Invisible for both Camera and LiDAR: Security of Multi-Sensor Fusion based Perception in Autonomous Driving Under Physical-World Attacks

Yulong Cao[⋆,§] Ningfei Wang[⋆,†] Chaowei Xiao[⋆,‖,‡‡] Dawei Yang[⋆,§] Jin Fang[‡] Ruigang Yang[††]
Qi Alfred Chen[†] Mingyan Liu[§] Bo Li[¶]
[†]University of California, Irvine, {ningfei.wang, alfchen}@uci.edu
[§]University of Michigan, {yulongc, ydawei, mingyan}@umich.edu
[‖]NVIDIA Research [‡‡]Arizona State University [††]Inceptio
[‡]Baidu Research and National Engineering Laboratory of Deep Learning Technology and Application, China
[¶]University of Illinois at Urbana-Champaign, lbo@illinois.edu

2021 IEEE Symposium on Security and Privacy (SP) | 978-1-7281-8934-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/SP40001.2021.00076

***Abstract*—In Autonomous Driving (AD) systems, perception is both security and safety critical. Despite various prior studies on its security issues, *all* of them only consider attacks on camera- or LiDAR-based AD perception *alone*. However, production AD systems today predominantly adopt a Multi-Sensor Fusion (MSF) based design, which in principle can be more robust against these attacks under the assumption that not all fusion sources are (or can be) attacked at the same time. In this paper, we present the first study of security issues of MSF-based perception in AD systems. We directly challenge the basic MSF design assumption above by exploring the possibility of attacking *all* fusion sources simultaneously. This allows us for the first time to understand how much security guarantee MSF can fundamentally provide as a general defense strategy for AD perception.**

We formulate the attack as an optimization problem to generate a physically-realizable, adversarial 3D-printed object that misleads an AD system to fail in detecting it and thus crash into it. To systematically generate such a physical-world attack, we propose a novel attack pipeline that addresses two main design challenges: (1) non-differentiable target camera and LiDAR sensing systems, and (2) non-differentiable cell-level aggregated features popularly used in LiDAR-based AD perception. We evaluate our attack on MSF algorithms included in representative open-source industry-grade AD systems in real-world driving scenarios. Our results show that the attack achieves over 90% success rate across different object types and MSF algorithms. Our attack is also found stealthy, robust to victim positions, transferable across MSF algorithms, and physical-world realizable after being 3D-printed and captured by LiDAR and camera devices. To concretely assess the end-to-end safety impact, we further perform simulation evaluation and show that it can cause a 100% vehicle collision rate for an industry-grade AD system. We also evaluate and discuss defense strategies.

I. Introduction

Today, high-level (e.g., Level-4 [1]) self-driving cars, or Autonomous Vehicles (AV) [2], are under rapid development. Some of them, e.g., Google Waymo [3] and TuSimple [4], are already providing services on public roads. To ensure correct and safe driving, a fundamental pillar in the Autonomous Driving (AD) system is *perception*, which leverages sensors such as cameras and LiDARs (Light Detection and Ranging) to detect surrounding obstacles in real time. Due to the direct impact on safety-critical driving decisions such as collision avoidance, various prior works have studied the security of AD perception under realistic physical-world attack vectors such as adding stickers, posters, or paintings to traffic signs [5]–[9], or shooting lasers to the LiDAR [10], [11].

All of these studies, however, are limited to attacks on a single source of AD perception, i.e., camera- or LiDAR-based AD perception *alone* [5]–[13]. By contrast, production high-level AD systems such as Waymo, Pony.ai, and Baidu Apollo, typically adopt a Multi-Sensor Fusion (MSF) based design [14]–[17], which fuses the results from different perception sources, e.g., LiDAR and camera, to achieve overall higher accuracy and robustness [18]–[26]. In such a design, under the assumption that not all perception sources are (or can be) attacked simultaneously, there *always* exists a possible MSF algorithm that can rely on the unattacked source(s) to detect or prevent such an attack. This basic security design assumption is believed to hold in general [27], [28], and MSF is thus widely recognized as a general defense strategy against existing attacks on AD perception [10], [27]–[29].

This paper presents a first study on the security property of MSF-based perception in AD systems today. We directly challenge the above basic security design assumption by demonstrating the possibility of effectively and simultaneously attacking *all* perception sources used in state-of-the-art MSF-based AD perception, i.e., camera and LiDAR [18]–[26]. This for the first time allows us to gain a concrete understanding of how much security guarantee the use of MSF can fundamentally provide as a general defense strategy for AD perception. Specifically, we consider physical-world attack vectors for high attack practicality, and target an attack goal with the most direct safety consequence for autonomous driving: cause a victim AV to fail in detecting a front obstacle.

Although prior works have designed successful physical-world attacks on AD perceptions based only on camera or only on LiDAR, we find that simply combining their designs does not achieve our goal. First, we need to identify a physical-

[⋆]Alphabetical ordering; The first four authors contributed equally.

DOI 10.1109/SP40001.2021.00076

world attack vector effective for both camera and LiDAR, which can not be satisfied by those popular ones used in prior works. For example, adding stickers changes an object's texture (e.g., color) but not its shape; this can be effective for camera [5]–[9] but not LiDAR. Conversely, laser shooting has been shown to be effective for LiDAR-based AD perception [10], [11], but not for camera-based ones. Second, no matter what attack vector we use, we need to further address 2 design challenges: (1) We need to differentiably synthesize the physical attack impacts simultaneously and consistently onto both camera images and LiDAR point clouds. For certain attack vectors, e.g., differentiably modelling the impact of lasers on camera images, this can be very challenging. (2) To improve run-time performance, the state-of-the-art LiDAR-based AD perception uses aggregated features of the 3D points grouped at the level of 2D or 3D cells [15], [20], [30]–[34]; however, their calculation is by nature non-differentiable (§III-B), which makes the attack difficult to optimize.

Towards this end, we design a novel physical-world adversarial attack method, MSF-ADV, which addresses the challenges above and thus fundamentally challenges the basic MSF design assumption in AD perception. We employ *adversarial 3D object* as the attack vector, with the key observation that different shapes of a 3D object can lead to both point position changes in LiDAR point clouds and pixel value changes in camera images. Thus, an attacker can leverage *shape manipulations* to introduce input perturbations to both camera and LiDAR simultaneously. To achieve the attack goal, the attacker simply places such an object on the roadway; this can be conveniently accomplished with modern 3D printing services and an object type commonly expected on the roadway, e.g., a traffic cone but with a slightly worn or broken look.

To systematically generate effective adversarial 3D objects, we adopt an optimization-based approach that starts with a 3D mesh of a normal object, e.g., a normal traffic cone, and performs shape manipulation by changing its vertex positions. Under this attack vector, we address design challenge 1 by constructing differentiable 3D rendering functions to synthesize the attack-influenced point clouds and camera images. For design challenge 2, we find that all commonly-used cell-level aggregated features can be differentiably derived by the *point-inclusion* property (§IV-D). Thus, we first design a differentiable and accurate approximation for such property, and then use it as a building block to differentiably compute the gradient of the cell-level aggregated features during the optimization. We also employ domain-specific designs for attack robustness, stealthiness, and physical-world realizability.

We evaluate MSF-ADV with MSF algorithms included in 2 open-source full-stack AD systems, Baidu Apollo [15] and Autoware.AI [16], that have high representativeness in practice, e.g., Apollo is ranked as the top 4 leading AD developers along with Waymo, Ford, and Cruise [35]. We select 3 object types and evaluate each on 100 real-world driving scenarios from the KITTI dataset [36]. Our results show that the generated adversarial objects achieve more than 91% success rate across different object types and MSF algorithms. We also find that our attack is (1) stealthy from the driver's view based on a user study, (2) robust to different victim approaching positions and angles, with over 95% average success rates, and (3) transferable across different MSF algorithms, with an average transfer attack success rate of around 75%.

To understand the attack realizability in the physical world, we 3D-print our adversarial objects, and evaluate them using real LiDAR and camera devices. Using a vehicle with a LiDAR mounted, we find that our 3D-printed adversarial object can successfully evade LiDAR detection in 99.1% (107) of the total 108 collected frames. Using a miniature-scale experiment setting (§V-E2), we find that our adversarial object has a 85-90% success rate to evade both LiDAR and camera detection at 20 randomly-sampled positions.

To understand the end-to-end safety impact, we further evaluate our method using a production-grade AD simulator, and find that our adversarial traffic cone can cause a 100% vehicle collision rate for an Apollo AV across 100 runs. In contrast, the collision rate with a normal traffic cone is 0%. Demo videos are at our project website: **https://sites.google.com/view/cav-sec/msf-adv**. We also evaluate various existing DNN-level defense strategies (e.g., input transformation and augmenting training data), and discuss future defense directions. Our code and data are released at our website [37].

In summary, this work makes the following contributions:

- We are the first to study security issues of MSF-based AD perception and the first to challenge the basic MSF design assumption in the AD context. We successfully design and engineer a physical-world adversarial attack aiming at generating adversarial 3D object to mislead a victim AV to fail in detecting it and thus crash into it.
- We adopt an optimization-based approach that addresses two main design challenges: non-differentiable target camera and LiDAR sensing systems, and non-differentiable cell-level aggregated features used by LiDAR. We also design strategies to enhance the attack robustness, stealthiness, and physical-world realizability.
- We evaluate on MSF algorithms included in representative open-source industry-grade AD systems in real-world driving scenarios. Our attack is shown to achieve over 91% success rates across different object types and MSF algorithms. Such high effectiveness can also be achieved with (1) high stealthiness, (2) high robustness to victim positions, (3) high transferability across MSF algorithms, and (4) high physical-world realizability after being 3D-printed and captured by LiDAR and camera devices.
- To understand the end-to-end safety impact, we further evaluate the proposed attack on a production-grade simulator, and show that our attack can cause a 100% vehicle collision rate to an industry-grade AD system. We also evaluate and discuss defense strategies.

While MSF is widely recognized as a promising and general defense strategy for existing attacks on AD perception [10], [27]–[29], [38]–[43], prior works have neither studied the security of existing MSF algorithms in practical AD settings, nor made attempts to understand whether the very basic

security design assumption for MSF can fail. In this paper, we make the first attempt towards this direction, and we hope that our findings and insights can inspire more future research into this largely overlooked research perspective.

II. Background

A. MSF-based AD Perception

In high-level (e.g., Level 4 [1]) AD systems, *perception* is a critical module that detects surrounding objects in real time. Due to its direct impact on safety-critical driving decisions such as collision avoidance, AD perception in production high-level AD systems such as Google Waymo, Pony.ai, and Baidu Apollo predominantly adopts a Multi-Sensor Fusion (MSF) based design [14]–[17]. In this paper, we call such design *MSF-based AD perception*, or *MSF* for short. In this paper, we focus on MSF designed for *in-road obstacle detection*, e.g., front cars, which is the most basic task for AD perception.

MSF design principle and basic assumption. In MSF-based AD perception, the final object detection results are obtained by fusing multiple perception sources such as camera and LiDAR, with the goal of leveraging their strengths while compensating their weaknesses to achieve overall higher accuracy and robustness than those achievable by a single perception source [18]–[26]. For example, LiDAR is a ranging-based sensor by shooting lasers, which thus is more difficult to capture the texture information (e.g., color) of an object compared to cameras [18]. Camera images, on the other hand, cannot directly provide the depth information of an object [19], [25], which can be overcome by LiDARs. Thus, an MSF algorithm can be designed to leverage both the depth information from LiDAR point clouds and the texture information from camera images to achieve higher object detection performance than those using either camera or LiDAR alone [19], [20]. To achieve such overall higher accuracy and robustness, the basic design assumption is that *there generally exists at least one source that can provide the correct results*. In this paper, we are the first to challenge such assumption in the AD context.

Representative MSF algorithm design. In AD perception, state-of-the-art MSF algorithms predominately use 2 perception sources: camera and LiDAR [18]–[24], [26]. Fig. 1 shows an overview of a typical MSF-based AD perception design. In industry AD systems, before running the MSF, the raw camera and LiDAR inputs, i.e., camera images and LiDAR point clouds, are usually first pre-processed [15], [16] to prepare the camera- and LiDAR-side MSF inputs, which can improve the run-time algorithm performance (detailed later).

In the MSF algorithm, state-of-the-art designs predominantly adopt DNN networks to process the LiDAR-side and camera-side MSF inputs [15], [16], [18]–[24], [26], due to the recent superior performance of deep learning in object detection [44]. In this paper, we call them *LiDAR perception networks* and *camera perception networks* inside the MSF algorithm. Next, the processing results from these two networks are fused using (1) DNNs [18]–[24], or (2) hard-coded matching and prioritization rules [15], [16]. Rule-based fusion is usually a late fusion, i.e., applied to the end results of the two networks, while DNN-based one can be a late or early fusion, i.e., at the intermediate perception results, which can be fused more deeply and thus potentially lead to higher accuracy. Meanwhile, rule-based fusion has two unique benefits. First, it is more modular and thus can flexibility combine different camera and LiDAR perception models [45]. Second, it is easier to debug and interpret than DNNs [46], and also to hard-code safety rules and measures [45]. In our attack design later in §IV, we comprehensively consider both fusion designs.

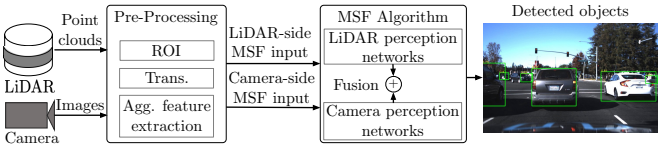


Figure 1: Overview of MSF-based AD perception design.

When preparing the camera- and LiDAR-side MSF inputs, typical pre-processing steps include data transformation such as rotations and shifting, applying Region of Interest (ROI) filter to remove unrelated input portions, and extracting aggregated features from the raw input. These pre-processing steps can largely reduce the sizes and dimensions of the MSF algorithm inputs, which can thus greatly improve the run-time algorithm performance [47]. Considering that the raw point cloud data can include millions of 3D points per second [48], such pre-processing is especially beneficial for LiDAR perception. Thus, many state-of-the-art LiDAR-based AD perception model designs choose to use aggregated input features such as average height and intensity of the 3D points grouped at the level of *3D cells*, or *voxels* [30]–[32], [49]. Some state-of-the-art designs even choose to further aggregate the features in such 3D cells to 2D cells in Bird's-Eye View (BEV) to further improve the real-time detection performance [34], which is thus the most popularly adopted in industry-grade AD systems [15], [20], [33], [34]. As detailed in §III-B, such popular adoption of cell-level aggregated features for LiDAR introduces a unique challenge to our attack design.

B. Physical-World Adversarial Attack

Recent works find that DNN models are generally vulnerable to adversarial example, or *adversarial attacks* [50]–[56]. Some works further explored such attacks in the physical world [5]–[9], [57]–[60]. In the AD context, previous works have designed successful physical-world adversarial attacks on the camera-based AD perception *alone* [5]–[9], [12], [13], or the LiDAR-based one *alone* [10], [11]. However, none of them have considered MSF-based AD perception, which is predominantly adopted in industry AD systems today (§II-A) and in principle can be more robust against these attacks (§I). Also, as detailed later in §III-B, blindly combining these prior designs cannot directly lead to successful attacks on MSF due to various new and unique challenges.

III. Problem Formulation and Design Challenges

A. Attack Goal and Threat Model

Attack goal: Fundamentally defeat MSF design assumption. In this paper, we target an attack goal with the most direct

safety impact on driving: fool the MSF-based AD perception in the victim AV to fail in detecting a front obstacle and thus crash into it. Even when the vehicle has a fail-safe Automatic Emergency Brake (AEB) system, e.g., based on RADAR or ultrasonic sensors, such a crash is still possible for two reasons. First, today's AEB systems are not perfect. For example, a recent study shows that the ones in popular vehicle models today fail to avoid crashes 60% of the time [61]. Second, even if they can successfully perform emergency stop, they cannot avoid being hit by rear vehicles that fail to yield on time. To achieve this goal, in this paper we target physical-world attack vectors in the AD context for high practicality and realism.

Due to the basic design assumption of MSF (§II-A), as long as there still exists at least one perception source that is not attacked, it is always possible for the unattacked source(s) to correct the final fused perception results and thus defeat our attack goal. Thus, in this paper we aim at designing an attack that can effectively attack *all* perception sources used in the MSF-based AD perception. This can enable our design to *fundamentally* defeat the MSF design assumption and thus most generally achieve our goal above. As the combination of camera and LiDAR is most popularly adopted in state-of-the-art MSF-based AD perception (§II-A), in this paper our design needs to attack both camera and LiDAR simultaneously.

Threat Model. As the first study to achieve the attack goal above, in this work we mainly focus on a white-box attack setting, i.e., assuming that the attacker has a full knowledge of the MSF algorithm used in the victim AD system. This is the same assumption made in most prior adversarial attacks on camera- or LiDAR-based AD perception [5], [6], [10], [62]. To achieve this, the attacker may obtain a victim AV model, e.g., by purchasing or renting [63], and then reverse engineer its perception module, which has been shown as possible on Tesla Autopilot [64]. The attacker can also target the AVs using open-source MSF-based AD perception algorithms [15], [16]. In the attack preparation time, we assume that the attacker can collect camera images and LiDAR point clouds of a targeted road where she plans to launch the attack.

B. Design Challenges

As described in §II-A, in state-of-the-art MSF algorithms, the camera and LiDAR perception networks are DNN based. Although no prior works consider attacking MSF, many designed successful physical-world adversarial attacks on camera- *or* LiDAR-based AD perception DNN models. However, we find that blindly combining these prior designs cannot directly achieve our goal due to 3 unique challenges:

C1. Lack of a single physical-world attack vector effective for both camera- and LiDAR-based AD perception. To achieve our attack goal, we need to find physical-world attack vectors for both camera- and LiDAR-based perception networks in MSF. However, so far none of the attack vectors used in previous physical-world adversarial attacks in the AD context have shown effectiveness in affecting both. For camera-based AD perception, previous works predominately consider adding stickers/posters [5], [6], painting [8], [9], or changing brightness [12], [13], which can only change the *texture* of an obstacle but not its *shape* and thus can barely affect the LiDAR point clouds. On the LiDAR side, LiDAR spoofing [10], [11], which shoots lasers to LiDAR, has shown to be effective in the AD context. Although lasers can also affect camera inputs [40], no prior work has studied its effectiveness for fooling camera-based AD perception models. One possible solution is to use separate attack vectors for them, e.g., using stickers for camera and laser shooting for LiDAR. However, this not only adds up the attack deployment costs and thus lowers the realizability and stealthiness, but also requires precise synchronizations across the attack processes. Thus, it is highly desired to identify one single attack vector that can effectively attack both at the same time.

C2. Need to differentiably synthesize physically-consistent attack impacts onto both camera and LiDAR. To systematically generate adversarial inputs, prior works generally adopt optimization-based approaches, which have shown both high efficiency and effectiveness [6], [60]. Since adversarial attack generation typically takes thousands of optimization iterations [65], [66], it is almost impossible in practice to physically drive vehicles on the target road to obtain the attack-influenced camera images and LiDAR point clouds every time the adversarial inputs are updated in an iteration. Thus, we need to digitally synthesize the impacts of the adversarial stimulus from the physical world onto both camera images and LiDAR point clouds, and such synthesizing needs to be differentiable to enable effective optimization. As discussed in **C1**, no single attack vector has been studied for both camera- and LiDAR-based AD perception so far in prior works. Thus, no matter what attack vector we identify to address **C1**, we need to design a new differentiable synthesizing function for at least one of the perception sources, which can be quite challenging for certain physical-world attack vectors, e.g., differentiably modelling the impact of lasers on camera inputs from different distances and angles. Meanwhile, since such attack impacts come from the same physical-world stimulus, the synthesized impacts to the camera images and the LiDAR point clouds need to be *physically consistent*, e.g., conforming to their different mounting positions in the AV.

C3. Need to handle non-differentiable pre-processing steps in AD perception. As introduced in §II-A, in industry AD systems, images and point clouds are usually pre-processed before fed into the MSF algorithm. In particular, state-of-the-art LiDAR-based AD perception models popularly use aggregated features of 3D points grouped at level of 2D or 3D cells (§II-A). To calculate such cell-level aggregated features, the necessary first step is to calculate whether an input point is inside a cell or not. In this paper, we call it a *point-inclusion* property. By nature, such property is discontinuous, i.e., 0 and 1 for outside and inside a cell. This causes the calculation of any cell-level aggregated features non-differentiable with regard to the LiDAR point clouds, which thus makes our optimization difficult to be effective. So far, no prior works have considered a general design to handle such non-differentiable pre-processing steps for LiDAR.

IV. Attack Design: MSF-ADV

In this paper, we are the first to address all the 3 challenges in §III-B by designing a novel physical-world adversarial attack method, *MSF-ADV*, which thus can fundamentally defeat the MSF design assumption in AD perception.

A. Design Overview

To address the challenges in §III-B, our MSF-ADV method has the following novel designs:

Adversarial 3D object: physically-realizable and stealthy attack vector for MSF-based AD perception. To address **C1**, we identify *adversarial 3D object* as the physical-world attack vector against MSF-based AD perception. Our key insight is that different shapes of a 3D object can lead to not only point position changes in LiDAR point clouds but also pixel value changes in camera images. Thus, the attacker can leverage *shape manipulations* of such an object to introduce adversarial input perturbations simultaneously to both camera and LiDAR perception networks in the MSF algorithm. To achieve the attack goal, the attacker simply places such an object in the roadway to trick the victim AV to crash into it.

Beside satisfying **C1**, such an attack vector also has 2 other advantages. First, it is easily realizable and deployable in the physical world. For example, the attacker can construct it digitally in a 3D mesh and 3D-print it, which is convenient today through online services [67]. Second, it can achieve high stealthiness by mimicking a normal traffic object that can legitimately appear in the middle of the road, e.g., a traffic cone or barrier, but with a worn or broken look, which is not uncommon in the real world as shown in Fig. 2. In our design (§IV-E), we also constrain the degree of the shape changes from the normal object to achieve high stealthiness. Note that although it is possible to manipulate the object texture (e.g., color) together with the shape in our design, we intentionally choose to not consider it in this paper as it can greatly harm stealthiness and also incur additional printability issues, which is a common challenge for physical-world adversarial attacks using stickers/posters [5], [68], [69].

Causing road safety threats. To make such an object both easy to deploy and able to cause severe crashes, the attacker can choose smaller objects such as a rock or traffic cone but fill it with granite or even metal to make it harder and heavier. For example, a 0.5 cubic-meter rock or a 1-meter high traffic cone [70] filled with some aluminum can easily weigh over 100 kg, which can trip the victim AV to lose control, damage the chassis, or break the windshield glass if bounced up when driving at a high speed. Besides causing damages by the crash itself, the attackers can also exploit the *semantic meaning* of certain road object types such as traffic cones. For example, the attacker can design an AV-specific attack by placing nails or glass debris behind an adversarial traffic cone object so that failing to detect it can lead to tire blowout of a targeted AV. Here, the safety damages are not directly caused by the traffic cone crash itself, and thus in this case the adversarial traffic cone can be small and lightweight like normal ones to make it easier to 3D-print, carry, and deploy.

Figure 2: Real-world traffic objects with worn or broken looking shapes, which can be mimicked by our physical-world attack vector: adversarial 3D object with *shape manipulations*.

Optimization-based adversarial 3D object generation. To systematically generate adversarial 3D objects, we adopt an optimization-based approach similar to prior works [5]–[11]. We start with a 3D mesh of a normal 3D object, e.g., a normal traffic cone, and then introduce shape manipulations by changing its vertex positions. To address **C2**, due to the choice of adversarial 3D objects as the attack vector, we can conveniently leverage existing 3D rendering techniques in computer graphics to simulate the functionalities of the physical equipment, i.e., camera and LiDAR, and thus systematically synthesize the attack-influenced camera images and LiDAR point clouds. Specifically, to enable the end-to-end optimization process, we perform differentiable constructions of these rendering processes, and use the relative positions to the 3D object to ensure the physical consistency with the corresponding camera and LiDAR mounting positions.

With the synthesized raw camera images and LiDAR point could, next we design the differentiable approximation function for the non-differentiable pre-processing step (non-differentiable cell-level aggregated feature calculation) to enable the end-to-end optimization. To address this, our key insight is that all the commonly-used cell-level aggregated features can be differentiably derived by the point-inclusion property (detailed later in §IV-D). Thus, we first design a novel and accurate differentiable function to approximate the calculation of the point-inclusion property, and then use it as a building block to achieve differentiable computations of the pre-processing steps for LiDAR. In the optimization process, we also have other domain-specific designs, e.g., for attack robustness, stealthiness, and physical-world realizability, which will be detailed in the following sections.

B. MSF-ADV Methodology Overview

In this section, we provide an overview of our MSF-ADV method, and will detail its components in later sections.

Problem formulation. We formulate the attack generation process as the following optimization problem:

$$\min_{S^a} \quad \mathbb{E}_{t\sim T}[\mathcal{L}_a(t(S^a);\mathcal{R}^l,\mathcal{R}^c,\mathcal{P},\mathcal{M}) + \lambda\cdot\mathcal{L}_r(S^a,\ S)] \tag{1}$$

$$\text{where} \quad \mathrm{PC}^a = \mathcal{R}^l(t(S^a),\ \mathrm{PC}) \tag{2}$$

$$\mathrm{IMG}^a = \mathcal{R}^c(t(\ S^a),\ \mathrm{IMG},\mathrm{C}) \tag{3}$$

$$F^a = \mathcal{P}(\mathrm{PC}^a,\ \mathrm{IMG}^a) \tag{4}$$

$$\mathcal{L}_a(t(S^a);\mathcal{R}^l,\mathcal{R}^c,\mathcal{P},\mathcal{M}) = \mathcal{O}(\mathcal{M}(F^a)) \tag{5}$$

$$\text{subject to} \quad \Delta(S^a,\ S) \le \epsilon \tag{6}$$

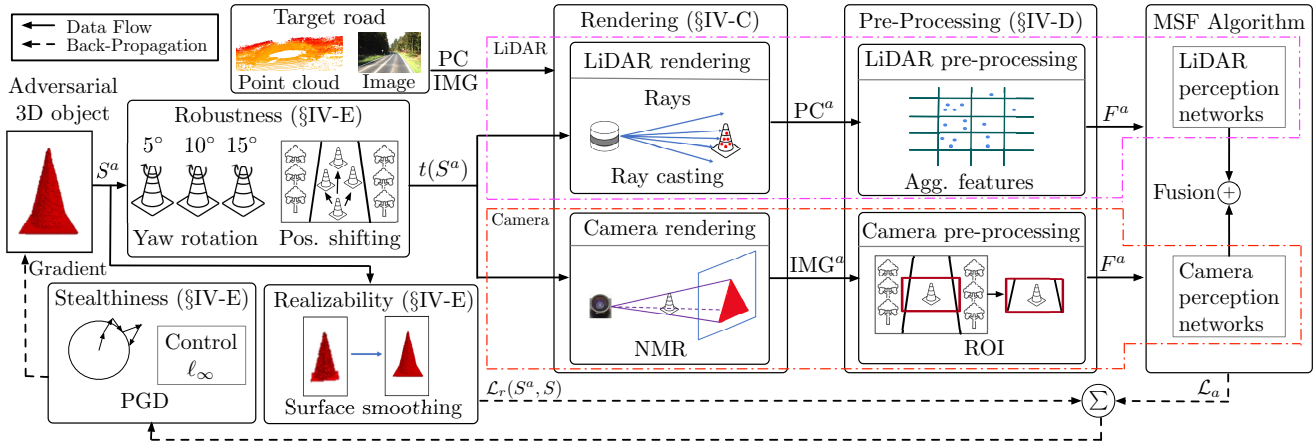


Figure 3: Overview of the optimization-based adversarial 3D object generation in MSF-ADV.

S is the original benign object and S^a is the adversarial one. We use vertex-face ($\boldsymbol{v}$-$\boldsymbol{f}$) meshes to represent them, i.e., $S = (\boldsymbol{v}, \boldsymbol{f})$ and $S^a = (\boldsymbol{v}^a, \boldsymbol{f}^a)$. In Eq. (1), the optimizing parameter is the adversarial object S^a, and we only change its vertices $\boldsymbol{v}^a$. The objective function includes: (1) an adversarial loss $\mathcal{L}_a$, which is designed to achieve our attack goal by misleading the MSF algorithm $\mathcal{M}(\cdot)$ to fail in detecting S^a, and (2) a realizability loss $\mathcal{L}_r(\cdot)$, which is designed to improve smoothness of the S^a surface to benefit both the printability and stealthiness (§IV-E). To improve the robustness of S^a in the physical world, we apply Expectation over Transformation (EoT) [60] by introducing a set of 3D transformation T to S^a and optimize the expectation of their objective function values in Eq. (1). λ is a balancing hyper-parameter.

In Eq. (2) and Eq. (3), $\mathcal{R}^l(\cdot)$ and $\mathcal{R}^c(\cdot)$ are the differentiable LiDAR and camera rendering functions respectively (§IV-C). They generate the attack-influenced point clouds PC^a and images IMG^a given the corresponding backgrounds of the target road (PC and IMG). PC^a and IMG^a are then fed into the differentiable pre-processing approximation function $\mathcal{P}(\cdot)$ to obtain the attack-influenced MSF input features F^a (§IV-D). F^a is fed into MSF algorithm $\mathcal{M}(\cdot)$ in Eq. (5), and $\mathcal{O}(\cdot)$ is designed to extract the output features related to the object's confidence score of the adversarial object. To achieve high stealthiness, in Eq. (6) we limit the shape deformation between S and S^a within a threshold ϵ by using a distance metric $\Delta(\cdot)$ (e.g., L_p distance metric: $\Delta(S^a, S) = ||S^a - S||_p$).

Optimization process overview. Fig. 3 overviews our optimization process. As shown, given a 3D object S^a initialized with S, we first apply 3D transformations (e.g. rotation and position shifting) T to generate multiple samples $t(S)$ to improve the robustness of the adversarial object against environment's variation. Next, each one of them, along with the LiDAR point clouds (PC) and camera image (IMG) background from the target road, are fed into the rendering functions ($\mathcal{R}^l(\cdot), \mathcal{R}^c(\cdot)$), pre-processing approximation functions ($\mathcal{P}(\cdot)$), and the MSF algorithm ($\mathcal{M}(\cdot)$) to calculate $\mathcal{L}_a$. Additionally, the realizability loss $\mathcal{L}_r(S^a, S)$ is added to $\mathcal{L}_a(\cdot)$ together using Eq. (1) to construct our loss function. To solve it, we use Projected Gradient Descent (PGD). Specifically, we compute its gradients with respect to the vertex positions $\boldsymbol{v}^a$ of S^a and constrain the gradients with a stealthiness threshold ϵ. We then update S^a using these gradients. We iteratively apply this process until S^a cannot be detected by the MSF algorithm.

Cell-level Aggregated Features	Used in
Occupancy	[15], [16], [30], [34], [49], [73]
Count	[15], [16]
Height (min/max/mean)	[15], [16], [20], [24], [33]
Intensity (min/max/mean)	[15], [16], [20], [30], [33], [34], [49]
Density	[20], [24], [33]

Table I. Summary of commonly-used cell-level aggregated features in state-of-the-art LiDAR-based object detection models. Our novel soft point-inclusion property calculation (§IV-D) can be used to differentiably derive all of them.

C. Differentiable Rendering

In this section, we detail the differentiable rendering functions $\mathcal{R}^l(\cdot)$ and $\mathcal{R}^c(\cdot)$ in Eq. (2) and Eq. (3). To ensure physical consistency, we define S^a in the LiDAR coordinate system, which is convenient as it is by nature 3D. For camera rendering, we then use a calibration matrix C to transform S^a from the LiDAR coordinate system to the camera coordinate system. C can be obtained by measuring the relative positions between the camera and the LiDAR of AV. To achieve differential rendering, we leverage existing differentiable ray-casting methods [71] for LiDAR and NMR [72] for camera.

D. Pre-Processing Step Approximation

In this section, we detail the construction of the differentiable pre-processing function $\mathcal{P}(\cdot)$. Most of the pre-processing steps such as ROI, rotation, and position shifting (§II-A) can be directly constructed differentiably using projective and affine transformations. However, such construction is especially challenging for the calculation of cell-level aggregated features such as cell occupancy and the mean height of the points inside a cell, due to the discontinuity of the point-inclusion property as discussed in **C3** (§III-B). However, such features are commonly used in state-of-the-art LiDAR-based AD perception as summarized in Table I, for achieving high

run-time performance (§II-A). This thus makes it necessary to address this to ensure the generality of our attack method.

To address this, we find that as long as we can obtain the point-inclusion value of each 3D point to a given cell, all the commonly-used features in Table I can be mathematically calculated in closed form. Thus, we first design an accurate and differentiable approximation of the point-inclusion property calculation, or a *soft* point-inclusion calculation, and then use it as a *building block* to differentiably derive the features.

Building block: Soft point-inclusion calculation. Given a point $\mathbf{PC}_i$ with coordinate (u_i, v_i, w_i) from the point cloud PC and a 3D cell c_m of length L, width W, and height H, the direct point-inclusion value of $\mathbf{PC}_i$ for c_m, denoted as $\mathrm{PI}(\mathbf{PC}_i, c_m)$, is 1 if $\mathbf{PC}_i$ is inside c_m, and 0 if not. To differentiably approximate this function, we estimate the *point-inclusion probability* of the point among the 8 cells closest to it by calculating the interpolation of it to these 8 cell center positions. Fig. 4 (a) illustrates these 8 cells, which are indexed as $m = 1...8$. The center position of a cell c_m is denoted as (u_m, v_m, w_m). These 8 center positions form a cuboid that encloses $\mathbf{PC}_i$. We can then calculate the interpolation of this point to these center positions using trilinear interpolation [74]:

$$\begin{aligned}\mathrm{softPI}(\mathbf{PC}_i, c_m) =&(1 - \frac{d(u_m, u_i)}{L}) \cdot (1 - \frac{d(v_m, v_i)}{W}) \\ &\cdot (1 - \frac{d(w_m, w_i)}{H})\end{aligned} \tag{7}$$

where $d(u_1, u_2) = |u_1 - u_2|$ and $\sum_{m=1}^{8} \mathrm{softPI}(\mathrm{PC}_i, c_m) = 1$. Thus, this is similar to calculating the probabilities of whether PC_i is inside each of these 8 cells. Fig. 4 (b) illustrates the calculation process and the example calculation for $\mathbf{PC}_i$ at $(0.8, 0.7, 0.1)$ when $L = W = H = 1$ (i.e., each cell is a cube) and the center coordinate of c_5 is the origin $(0, 0, 0)$. The calculation results are the numbers without underline at the 8 center positions. In Fig. 4 (c), the interpolation value at the center position of each cell is then used as the point-inclusion probability for such cell. As shown, since $\mathbf{PC}_i$ is inside c_7, it is the closest to the center of c_7 at $(1, 1, 0)$, and thus the interpolation value is the highest for c_7. This thus is able to correctly assign the highest point-inclusion probability to c_7.

Approximation accuracy improvement. In Fig. 4 (b), while the point-inclusion probability is indeed the highest for c_7, the probability value is only 0.504 and thus still has a non-negligible gap to the ground-truth value 1. We find that the cause of this gap is at the $d(u_1, u_2)$ function in Eq. (7). As shown in Fig. 5, the ground-truth function for $d(u_1, u_2)$ when $L = W = H = 1$ is $0.5 + 0.5 \cdot \mathrm{sign}(|u_1 - u_2| - 0.5)$, since if the distance between the point and the cell center at any dimension is over 0.5, it is outside of cell and thus $(1 - d(u_1, u_2))$ should be 0 in Eq. (7). Since $\mathrm{sign}(x)$ is not differentiable when $x = 0$, such ground-truth function cannot be directly used in $\mathrm{softPI}(\cdot)$. Using $d(u_1, u_2) = |u_1 - u_2|$ as in classic trilinear interpolation is differentiable, but its curve has a gap to the ground truth as shown in Fig. 5 so that it is more difficult for the optimized S^a to succeed. To address this, we use $\tanh(\cdot)$ to differentiably and accurately approximate $\mathrm{sign}(\cdot)$. For example, for the u dimension, it becomes:

$$d(u_1, u_2) = \frac{L}{2} + \frac{L}{2} \cdot \tanh(\mu \cdot (\mid u_1 - u_2 \mid - \frac{L}{2})) \tag{8}$$

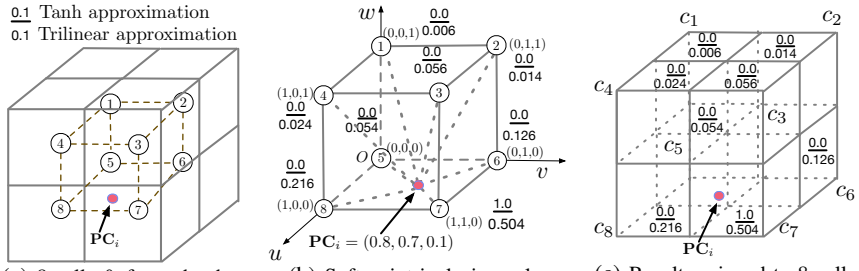
For the v and w dimensions of $\mathbf{PC}_i$ we replace L with W and H. Fig. 5 shows the curve of Eq. (8) when $L = 1$. As shown, the difference between Eq. (8) approximation and the ground truth is much smaller. In this paper, we call $\mathrm{SoftPI}(\cdot)$ using $d(u_1, u_2) = |u_1 - u_2|$ and Eq. (8) *trilinear* and *tanh approximation* respectively. The numbers with underline in Fig. 4 (b) and (c) are the results with tanh approximation. As shown, with tanh approximation the point-inclusion probability for c_7 becomes 1.0, which is directly the ground-truth value and thus much more accurate than trilinear approximation.

To more concretely show the benefit of tanh approximation, Fig. 6 shows the calculation results for the count feature in Table I based on $\mathrm{softPI}(\cdot)$ using real-world point cloud data. The count feature calculates the number of points in a cell (derivation of it from $\mathrm{softPI}(\cdot)$ is described later). In Fig. 6, the count values are visualized using a gray-scale heatmap in BEV. Fig. 6 (a) and (c) shows the count values calculated using trilinear and $\tanh$ approximations respectively, and (b) and (d) shows their differences to the ground-truth count value. As shown, the count values using trilinear approximation have clear differences to the groundtruth, while the differences for the ones using tanh approximation is almost invisible.

Derivation of cell-level aggregated features. With an accurate $\mathrm{SoftPI}(\cdot)$, we can then differentiably approximate all the cell-level aggregated features in Table I as follows:

- *Count and density.* The count feature calculates the number of points in a cell. With $\mathrm{softPI}(\cdot)$, we can differentiably derive the count value as $\mathrm{CNT}(c_m) = \sum_{\mathbf{PC}_i \in \mathrm{PC}} \mathrm{softPI}(\mathbf{PC}_i, c_m)$. The density feature calculates the density of points in a cell. Thus, we can directly calculate it by dividing $\mathrm{CNT}(\cdot)$ by the cell size.
- *Occupancy.* The occupancy feature calculates whether a cell has points or not. With $\mathrm{CNT}(\cdot)$ above, it can be calculated as $\mathrm{sign}(\mathrm{CNT}(\cdot))$. Note that since the $\mathrm{sign}(\cdot)$ is not differentiable, we approximate it using $\mathrm{sign}(x) = x$ during the backward pass of the optimization.
- *Height and intensity.* The max/min/mean height features calculate the maximum, minimum, and the average height of the points inside a cell. Thus, the max and min height features are directly $\max_{\mathbf{PC}_i \in \mathrm{PC}} \mathrm{softPI}(\mathbf{PC}_i, c_m) \cdot w_i$ and $\min_{\mathbf{PC}_i \in \mathrm{PC}} \mathrm{softPI}(\mathbf{PC}_i, c_m) \cdot w_i$. The mean height feature can be calculated as $\frac{\sum_{\mathbf{PC}_i \in \mathrm{PC}} \mathrm{softPI}(\mathbf{PC}_i, c_m) \cdot w_i}{\mathrm{CNT}(c_m) + \epsilon}$, where ϵ is small number to prevent division by zero. The max/min/mean intensity features can be calculated similarly by replacing w_i with the intensity value of $\mathbf{PC}_i$.

The calculations above are performed for 3D cells. To obtain features for 2D cells, we just need to add an aggregation of these 3D cell features in one dimension, e.g., the vertical dimension for BEV 2D cells (§II-A), into these calculations.



(a) 8 cells & formed cube (b) Soft point-inclusion calc. (c) Result assigned to 8 cells

Figure 4: Illustration of the soft point-inclusion calculation with trilinear and tanh approximations. $\mathbf{PC}_i$ is a point in PC, and c_1 to c_8 are the 8 3D cells closest to $\mathbf{PC}_i$.

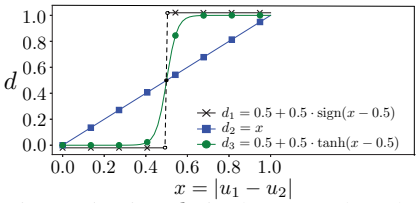


Figure 5: Line d_1 is the ground truth, while d_2 and d_3 are trilinear and tanh approximation. As shown, the tanh one is much closer to the ground truth.

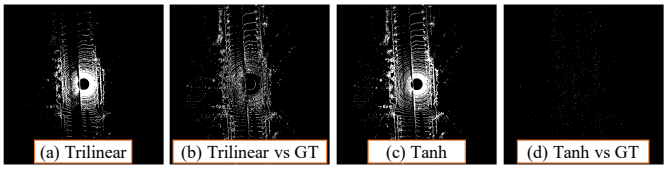


Figure 6: Accuracy benefit of tanh approximation over trilinear approximation for the count feature (number of points per cell). Count values are visualized using a gray-scale heatmap. GT denotes the ground-truth count value.

E. Objective Function Design

Adversarial loss $\mathcal{L}_a$. For the adversarial loss $\mathcal{L}_a$ in Eq. (1), similar to prior attacks on object detection [5], [6], we extract and minimize the confidence value (which reflects the confidence that the region contains an object) of the regions of S^a. As introduced in §II-A, the fusion process of the LiDAR and camera perception networks in the MSF algorithm can be DNN-based or rule-based. For the former, we directly extract the confidence values in the MSF output [18]–[24]. For the latter, since the rule-based fusion logic is not directly differentiable, we extract the confidence values in the outputs of the LiDAR and camera perception networks separately, and minimize the sum of them. This is because if we can prevent S^a from being detected in the outputs of both the LiDAR and camera perception networks, S^a will not appear in the MSF output no matter what the rule-based logic is.

Realizability loss $\mathcal{L}_r(\cdot)$. To realize our attack goal in §III-A, S^a needs to be 3D-printed and placed on top of the road surface in the physical world. To facilitate this, we design the realizability loss $\mathcal{L}_r(\cdot)$ in our objective function to (1) improve the printability of S^a at 3D printers by maximizing its surface smoothness using a Laplacian loss [75], and (2) prevent the generation of S^a that is underneath the road surface. The detailed loss formulations are in Appendix A.

Stealthiness designs. Our optimization process has two designs for improving the stealthiness of S^a. First, the realizability loss above can improve its surface smoothness, which can thus allow it to look normally in practice. Second, we solve Eq. (6) by using Project Gradient Descent (PGD) with L_∞ distance constraint during the gradient update step in Fig. 3, which thus ensures that the per-dimension moving distance for each vertex in S is smaller than ϵ. We can then use ϵ to control how similar S^a looks compared to the benign one S, and thus the smaller ϵ is, the stealthier S^a is.

Attack robustness improvement. To achieve the end-to-end attack success in our setting, it is ideal if S^a can be continuously undetected by the MSF algorithm when the victim AV is approaching the object, until their distance is smaller than the brake distance [76] so that it is too late to brake to avoid the crash. Thus, we need to improve the robustness of S^a against different victim approaching distances and angles of the target road. To achieve this, we implement Transformation T via random yaw-dimension rotations and ground-plane position shifting of S^a, which is illustrated in Fig. 3.

V. Attack Evaluation

A. Evaluation Methodology and Setup

MSF algorithm selection. In our evaluation, we target MSF algorithms included in open-source industry-grade AD systems to ensure high practicality and realism of our evaluation results. In particular, we select the ones included in 2 open-source full-stack AD systems, Baidu Apollo [15] and Autoware.AI [16], due to their (1) *representativeness* among industry-grade AD systems today, as Apollo has been recently ranked among the top 4 leading industrial AD developers along with Waymo, Ford, and Cruise [35], and Autoware is adopted by the USDOT in their AV fleet [77]; (2) *practicality*, since both systems can be readily installed on real vehicle models [78], [79] for driving on public roads. In particular, Apollo has been providing self-driving taxi services in China for months [80]; and (3) *ease to experiment with*, since they are the only full-stack AD systems that are open-sourced.

Both AD systems use rule-based fusion in their MSF algorithms, i.e., the LiDAR and camera perception networks are separated DNN models, and their individual perception outputs are fused based on hard-coded matching and prioritization rules. As described in §II-A, such design has high modularity and is easy to debug, interpret, and hard-code safety rules/measures [45]. These can greatly benefit system development in industry, which might be the reasons why it is adopted in both Apollo and Autoware.AI. As described in §IV-E, for such fusion type, our optimization objective is to make our adversarial object undetected in both the outputs of the LiDAR and camera perception models to allow our attack to succeed no matter what rule-based fusion logic is used.

Due to such modular fusion designs, the MSF algorithms in both Apollo and Autoware.AI allow different combinations of LiDAR and camera perception models. Thus, in our evaluation

we also evaluate our attack against different such combinations to understand the generality of our attack. In this paper, we call each such combination an *MSF combination* and use ⊕ to denote such combination operation. In particular, we select 2 different models for LiDAR and 2 for camera, which forms 4 MSF combinations in total. On the LiDAR side, the LiDAR perception model in Apollo is also included in Autoware.AI. Thus, we choose 2 models in different Apollo versions that have substantially different DNN designs: one from the latest version, v5.5, denoted as *A5-L*, and another from an older version, v2.5, denoted as *A2-L*. At the DNN design level, *A5-L* differs greatly from *A2-L* with 43.9% more deep layers and 65.0% more trainable parameters. On the camera side, we select the one from the latest version of Apollo, denoted as *A5-C*, and the pre-trained YOLO v3 [81], denote it as *Y3*, which is included in the latest version of Autoware.AI.

3D object type selection. Considering the supported object types for the LiDAR and camera models, we experiment with 3 types of objects for the above 4 MSF combinations: (1) a *traffic cone* of size 0.5 m × 0.5 m × 1.0 m, for A5-L⊕A5-C and A2-L⊕A5-C, (2) a *bench* of size 0.6 m × 0.5 m × 1.5 m, for A5-L⊕Y3 and A2-L⊕Y3, and (3) a *toy car* of size 0.6 m × 0.7 m × 1.6 m (for kids to sit inside), for all 4 MSF combinations. We intentionally avoid large objects like cars since they are much harder to 3D-print and deploy. Among the 3 object types, we consider traffic cone as the most attractive for attacker since it is much more common to appear on the roadway than the other two and thus the most stealthy. Thus, majority of our experiments are focused on traffic cone.

Attack scenario selection. For each object type, we select 100 real-world driving scenarios from the KITTI dataset [36] in which such object in benign case can be 100% detected by the MSF combinations. Each scenario is one frame of sensor inputs including the camera image, the LiDAR point cloud, and the calibration matrix. These scenarios has high diversity with different types of objects (e.g., cars, trucks, traffic lights) and roads (e.g., local, high-way, to rural roads).

Object placement. For most experiments, we place the benign and adversarial objects 7 meters (m) in front of the victim. We choose 7 m because it is the braking distance [76] when the vehicle speed is 25 mph, almost the lowest one in normal driving. Since such distance is larger for higher vehicle speeds, 7 m represents the *smallest* distance at which the object has to be detected by the victim to avoid a crash in normal driving scenarios. In §V-D, we also evaluate our attack among different victim distances and angles. More detailed attack parameter settings are in Table VIII in Appendix.

B. Attack Effectiveness

In this section, we evaluate the effectiveness of our attack on the attack scenarios described in §V-A.

Evaluation metrics. Given an MSF combination and an attack scenario, we render our generated 3D adversarial object into the background point cloud and image, and test whether it can be detected by the MSF combination. We determine our attack as success if and only if the adversarial 3D object is undetected by *both* the LiDAR and camera models in such MSF combination. Under this criterion, the successful attacks can generally defeat *any* rule-based fusion logic that can be applied to fuse the outputs of these two models. Thus, the calculated success rate is a *lower bound* when a specific fusion logic is used, e.g., the ones in Apollo and Autoware.AI. We perform evaluation on 100 scenarios and report success rate.

Results. The results for the 4 MSF combinations are shown in Table II. For all object types and all MSF combinations, success rates are at least 91%, and those for traffic cone and bench are all 100% among 100 driving scenarios. This shows that MSF-ADV is an effective method. Among these results, the 100% success rates for traffic cone is especially important, since it is the most attractive object type among the three from the attacker's view due to its small size and the ability to disguise as a normal traffic object in the middle of the road. Note that our method can achieve 91% attack success rates even for the toy car of which the object type (car) has been heavily explored in training data of the model. Among the 4 MSF combinations, A5-L⊕A5-C has the lowest attack success rates, which shows that the models from the latest version of Apollo are the most robust among the 4.

Stealthiness. We also measure the stealthiness of our object using the average per-vertex $\Delta\ell_p$ distances and the LPIPS (Learned Perceptual Image Patch Similarity) metric [82]. Table II shows the results with stealthiness parameter $\epsilon = 2$ cm (§IV-E). As shown, our attack only needs to move each vertex by 3.4 cm on average ($\Delta\ell_2$) to achieve at least 91% success rates on all MSF combinations. For LPIPS, we use the official implementation from [82] to measure the LPIPS value between the driving image with benign object rendered and the same image with the adversarial one rendered at the same location. As shown, the average LPIPS value is 0.10 across the 3 object types. This is at the same level as those achieved in latest GAN-based image restoring methods [83], which are generally considered as indistinguishable for human. In Table III, we further evaluate our attack under different ϵ values on A5-L⊕A5-C using traffic cone. As shown, the attack success rates are still over 93% even when the average moved distance per vertex ($\Delta\ell_2$) is as small as 1.5 cm.

Attack stealthiness user study. To more directly evaluate the attack stealthiness, we also conduct a user study for traffic cone with 105 participants from Amazon Mechanical Turk [84]. The results show that the generated adversarial traffic cone is generally viewed (1) as innocent as the original benign cone, and (2) less suspicious than certain benign ones with broken shapes. More details are in Appendix B.

Effectiveness under different attack settings. We also perform evaluation under different attack parameter settings. We find that our attack is most sensitive to μ, which show that the differentiable approximation design in §IV-D is critical to the attack success. More details are in Appendix C.

Printability. We also evaluate the printability of our attack using commercial printability checking tool and geometry metrics such as *watertightness* [85], [86], *self-intersection* [87], and *curvature* [87]. Our results show that our generated objects

MSF Comb.		A5-L⊕A5-C		A5-L⊕Y3		A2-L⊕A5-C		A2-L⊕Y3	
Object Type		Traffic cone	Toy car	Bench	Toy car	Traffic cone	Toy car	Bench	Toy car
Success Rate		100%	91%	100%	93%	100%	96%	100%	97%
Dist. (cm)	$\Delta\ell_1$	5.92	5.95	5.93	5.97	5.93	5.63	5.90	5.61
	$\Delta\ell_2$	3.28	3.46	3.39	3.37	3.43	3.34	3.30	3.25
	$\Delta\ell_\infty$	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
LPIPS		0.06	0.02	0.20	0.04	0.07	0.17	0.20	0.06

Table II. Attack success rate and average vertex perturbation distance of MSF-ADV on different MSF combinations in 100 driving scenarios. A5-L, A5-C: LiDAR and camera models in Baidu Apollo v5.5. A2-L: LiDAR model in Apollo v2.5. Y3: YOLO v3. All objects can be 100% detected by each MSF combination in the benign case.

Stealthiness Level (cm)	Succ. Rate	$\Delta\ell_p$ Dist. (cm)			LPIPS
		$\Delta\ell_1$	$\Delta\ell_2$	$\Delta\ell_\infty$	
$\epsilon = 2.0$	100%	5.92	3.28	2.00	0.06
$\epsilon = 1.0$	93%	2.84	1.51	1.00	0.05
$\epsilon = 0.5$	76%	1.38	0.54	0.50	0.05

Table III. Stealthiness evaluation results of MSF-ADV on MSF combination A5-L⊕A5-C with the traffic cone object under different stealthiness levels of ϵ (§IV-E).

Attack Method	Success Rate	$\Delta\ell_p$ Dist. (cm)		
		$\Delta\ell_1$	$\Delta\ell_2$	$\Delta\ell_\infty$
GN	8%	21.8	3.35	10.3
GA	9%	2.85	1.84	2.00
Ours	100%	5.92	3.28	2.00

Table IV. Comparison between MSF-ADV and baseline attack methods in attack success rate and object perturbation degrees. GN: Gaussian noise. GA: genetic algorithm.

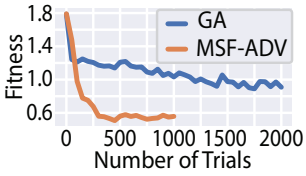


Figure 7. Fitness values of GA and MSF-ADV during the optimization process. The curve for MSF-ADV stops at 1000 since it can already succeed for all scenarios at that point.

are 100% printable, and our printability improvement designs in §IV-E substantially reduce the printing difficulties from 58.9% to 74.3%. Detailed are in Appendix D.

Transferability. We also evaluate the attack transferability among the 4 MSF combinations with the toy car object. We find that the transfer attack among them is generally effective, with success rates around 75% on average.

C. Comparison with Baseline Attack Methods

While our attack shows high effectiveness in the previous section, it is unclear how much of it is due to the specific designs in MSF-ADV. To understand this, in this section we compare our method with possible baseline attack methods.

Evaluation methodology. We consider 2 baseline attack methods: (1) Gaussian noise based shape perturbation, denoted as *GN*, and (2) Genetic algorithm [88] based attack generation, denoted as *GA*. GN is used to understand whether the success of our attack is due to our optimization-based design (§IV), or simply due to the nature of that level of shape perturbations. GA still uses our objective function design in §IV-E as fitness function, but does not need differentiability, which is thus used to understand whether our differentiable approximation function designs in §IV-D are actually useful.

Experimental setup. We perform comparison with our attack on A5-L⊕A5-C MSF combination with the traffic cone object using the same setup in §V-A. We implement GN and GA using the corresponding standard Python libraries [89], [90]. For GN, we apply a Gaussian noise with $\mu = 0$ and $\sigma = 2.1$ cm to each vertex dimension to generate a similar level of perturbation as MSF-ADV with ϵ = 2 cm. For GA, we set the population size to 50, a common value used in genetic algorithm based adversarial attacks [91], [92]. We configure it to use 2 cm as the per-dimension perturbation bound for each vertex, the same as ϵ in MSF-ADV. To achieve a fair comparison, we run GA using similar CPU and GPU resources as MSF-ADV, and ensure that it runs longer than our method.

Result. Table IV summarizes the attack success rates of GN, GA, and our method, and the corresponding shape perturbation degrees. As shown, for GN, the average moved distance per vertex is 3.35 cm ($\Delta\ell_2$), which is larger than those generated by our method (3.28 cm). However, only 8% of the ones from GN succeed, which is a magnitude lower than ours (100%). This thus shows that our high attack effectiveness is mainly due to our optimization-based design, instead of the nature of a similar-level shape perturbation. For GA, we stop it after it generates 2000 adversarial objects for each attack scenario, which is twice the number for our method (1000). However, the success rate is only 9%, which is also a magnitude lower than ours. Fig. 7 shows the fitness value trend during the optimization process, which is averaged over the 100 attack scenarios. As shown, the fitness value decrease for GA is much slower than ours: its fitness value drop after 2000 trials is achieved after only 133 trials using our method, which is 15× more efficient. This thus concretely shows that benefit of our differentiable approximation function designs in §IV-D, which allows the use of gradient-based optimizations to significantly improve both the attack efficiency and effectiveness.

D. Attack Robustness

In this section, we evaluate our attack robustness against different victim approaching positions and angles.

Evaluation methodology. We still use the attack scenarios in §V-A for evaluation. To synthesize different relative positions between the victim and the object when the victim

	Y = (-0.1 m, 0.1 m)		
	X = (5 m, 15 m)	(15 m, 25 m)	(25 m, 35 m)
w/o EoT	80.3%	79.2%	79.9%
w/ EoT	96.3%	95.5%	96.6%

Table V. Average success rate on A5-L⊕A5-C with traffic cone in different victim approaching distance ranges.

is approaching the object, we render the object at different locations ahead of the victim in both the camera and LiDAR frames given an attack scenario.

Experimental setup. As described in our attack design (§IV-A), the adversarial object is placed in the middle of the traffic lane in which the victim is driving. In this section, X and Y denote the relative distance between the victim and the object in the longitudinal (i.e., forward and backward) and lateral (i.e., left and right) directions respectively. For X, we consider 3 distance ranges from 5 to 35 m, which correspond to the brake distances for speed from ∼20 to 55 mph [76]. For Y, the deviations to the center of the lane usually need to be within 0.1 m for smooth and safe driving [93], [94]. Thus, we consider $Y \in (-0.1\text{ m}, 0.1\text{ m})$. For each position range, we randomly sample 20 different positions.

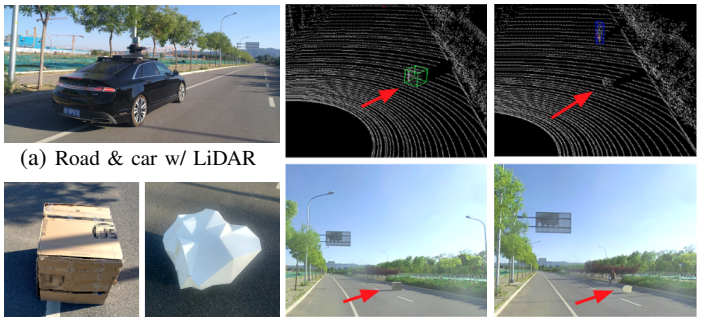
Results. Table V shows the average attack success rates for A5-L⊕A5-C with traffic cone in the 3 position ranges over the 100 evaluation scenarios (§V-A). As described in §IV-E, we use EoT to improve robustness. As shown, this improves the average success rates in all position ranges by 20.5% on average. Overall, with EoT the average attack success rates are over 95% across different position ranges, which shows a high robustness of our attack against different victim approaching positions and angles at common driving speeds.

E. Physical-World Attack Realizability Evaluation

While the results in prior sections show high effectiveness and robustness of our attack, the experiments are performed by digitally rendering the objects into camera and LiDAR inputs. Thus, it is unclear whether such high effectiveness can still be achieved after the adversarial object is 3D-printed and placed in the physical world. Thus, in this section we evaluate such physical-world realizabilty of our attack.

1) Real Vehicle based Experiments: At the early stage of this project, we had access to a real vehicle equipped with a high-end Velodyne HDL-64E LiDAR, and used it to perform physical-world experiments for LiDAR models. Unfortunately, later we lost the access to it and only have such real vehicle based experiments for the LiDAR-side evaluation. In this section, we report these results for LiDAR side, and will detail in the next section the physical-world experiments for both LiDAR and camera using a miniature-scale experiment setup.

Evaluation methodology and setup. In this experiment, we 3D-print the adversarial object and conduct the experiment by using the vehicle mentioned above to collect its LiDAR point clouds on the real road. Fig. 8 (a) shows the vehicle and road. We selected a rarely-used road and no other vehicles passed by during this experiment. Since this experiment was performed



(a) Road & car w/ LiDAR (b) Benign and adv. cubes (c) Benign case (d) Adversarial case

Figure 8: Physical-world experiment settings and evaluation results for LiDAR-side physical-world attack realizability. We use a Velodyne HDL-64E LiDAR mounted on a real vehicle. The adversarial cube is 3D-printed at 1:1 scale.

at the early stage of this project, the selected object type was a 75cm cube, and the targeted model was A2-L, the latest version of the Apollo LiDAR model at that time. Fig. 8 (b) shows the box of the same size used as the benign cube, and the 3D-printed adversarial cube. This setup mimics the attack scenario by placing an adversarial rock-shaped object (§IV-A).

Results. We manually drive the vehicle around the cube and collect traces in front of it and on the left of it. In total, there are 99 LiDAR frames with the benign cube, and A2-L is able to correctly detect it in 84.8% (84) frames. In comparison, we find that the adversarial cube is detected in only *0.9% (1)* of the 108 LiDAR frames including it. Fig. 8 (c) and Fig. 8 (d) show examples of the frames and detection results for the benign and adversarial cubes respectively. These results show that our attack is still effective in the physical-world setting for the LiDAR side of MSF. Experiment videos and images are at **https://sites.google.com/view/cav-sec/msf-adv**.

2) Miniature-Scale Experiments: Since we lost the access to the experiment vehicle, in this section we design a miniature-scale experiment in our lab environment to perform physical-world experiments for both LiDAR and camera.

Evaluation methodology. In this experiment, we still 3D-print the adversarial object and obtain its point clouds and images using physical LiDAR and camera devices like in the actual physical-world attack settings. However, the main difference is that the adversarial object and the road are set up in a *miniature scale* as shown in Fig. 9. As shown, the adversarial object is 3D-printed at 1:6.67 scale and placed on a miniature-scale straight road created by printing a real-world high-resolution BEV road texture on multiple A4 papers and concatenating them together. Here, the obtained point clouds of the object and road are scaled up accordingly following the physical rule of LiDAR to obtain the point clouds in real-world scale. The benefit of such miniature-scale setup is that it can not only obtain physical-world point clouds and images following the same physical rules of LiDAR and camera, but also more easily fit into the budget of a university-level research lab (e.g., 3D-printing our 1-meter high traffic cone at 1:1 scale requires industry-grade 3D printers [95]).

Experimental setup. We use an iPhone 8 Plus back camera

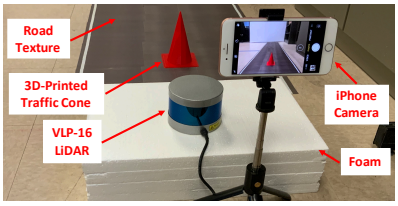


Figure 9: Miniature-scale experiment setup with camera and LiDAR. Road and traffic cone are at 1:6.67 scale.

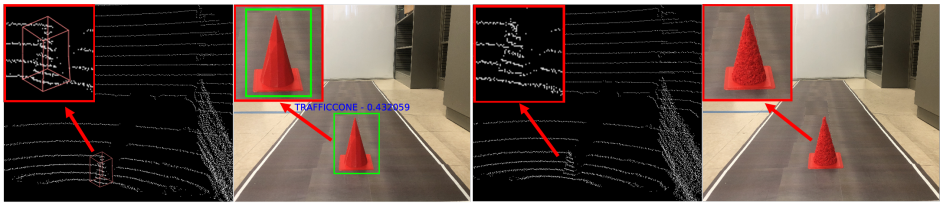


Figure 10: Visualization of the LiDAR and camera perception results for A5-L⊕A5-C in miniature-scale experiments.

	A5-L⊕A5-C	A2-L⊕A5-C (transfer attack)
Benign detection rate	19/20 (95%)	16/20 (80%)
Attack success rate	18/20 (90%)	17/20 (85%)
Attack success rate when benign can be detected	18/19 (94.7%)	14/16 (87.5%)

Table VI. Evaluation results for A5-L⊕A5-C and A2-L⊕A5-C at 20 randomly-sampled positions in miniature-scale experiments. Results for A2-L⊕A5-C is a transfer attack since the adversarial traffic cone is generated for A5-L⊕A5-C.

and a Velodyne VLP-16 LiDAR to collect images and point clouds as shown in Fig. 9. For the adversarial object, we generate the adversarial traffic cone mesh using the image and point cloud collected in our miniature-scale setup as the background. We 3D-print the benign and adversarial traffic cones with 380 um precision at 1:6.67 scale. The road size, traffic cone size, and the camera and LiDAR positions are chosen to represent the scenario where these sensors are installed on a car driving on a standard 3.6-meter wide highway road [96].

In the experiment, we try 20 different positions on the miniature road, which are randomly sampled in a 6.0 cm $\times$ 6.0 cm area at the road center and $\sim$45 cm far from the camera and LiDAR. We choose this area because we find the highest detection rate of the benign cone in this area. In real-world scale, this represents the scenario where the adversarial cone is roughly at the road center and 3-3.5 m far from the camera and LiDAR on the victim. Since the object type is traffic cone, we consider A5-C on camera side, and the VLP-16 versions of A5-L and A2-L in Apollo, which has the same model architecture as their corresponding HDL-64 versions [97].

Results. Table VI shows the results. As shown, for A5-L⊕A5-C, the benign traffic cone can achieve 95% detection rate at the 20 random positions. However, after we place the adversarial one at exactly these 20 positions, the detection rate is only 10%, leading to a 90% success rate. Specifically, at the 19 positions that the benign cone can be successfully detected, the attack success rate is around 95%. Fig. 10 visualizes the LiDAR and camera perception results of the benign and adversarial cones. More images and dynamic moving videos are at **https://sites.google.com/view/cav-sec/msf-adv**.

Since this adversarial cone is generated for A5-L⊕A5-C, we also evaluate it against A2-L⊕A5-C to understand whether such attack effectiveness can transfer. As shown, the success rate of such a transfer attack is very similar: the success rate among the 20 positions is 85%, and that among the positions where the benign cone can be detected is 87.5%. These results thus show that our generated adversarial objects can still be effective against both LiDAR and camera in a physical-world environment, and such effectiveness can transfer.

VI. End-to-End Attack Simulation Evaluation

To more concretely understand the end-to-end safety consequences, we further evaluate on a concrete attack scenario using a production-grade AD simulator.

Evaluation methodology and metrics. We perform an end-to-end attack evaluation on Baidu Apollo using LGSVL simulator [98]. LGSVL is an open-source Unity-based simulator designed for testing and development of industry-grade AD systems, and has already supported Apollo. In our evaluation, we use a map of a single-lane road in LGSVL, and set up Apollo to control a vehicle to drive along this lane. To launch our attack, we imported the 3D mesh of our adversarial traffic cone into Unity, set its physical properties, and then re-build the simulator and the map. We control the position of this adversarial cone to set it to the lane center, and LGSVL will provide Apollo with the raw camera and LiDAR inputs with the adversarial objects using its simulation engine. As described in §IV-A, crashing into such an adversarial traffic cone can lead to severe safety damages as the attacker can fill it with denser materials such as granite or metal, or put nails or glass debris behind it. Considering such concrete attack scenarios, we directly use the vehicle collision rate with the adversarial cone to evaluate the attack effectiveness.

Experimental setup. We evaluate on Apollo v5.0, the latest Apollo version supported by LGSVL so far [98]. We use the default camera and LiDAR device configurations in this support. The LiDAR and camera models in Apollo v5.0 are the same as those in the latest version, Apollo v5.5. Thus, we directly use the adversarial traffic cone generated in §V for this evaluation. The vehicle speed is set to 30 km/h. For both benign and adversarial scenarios, we perform 100 runs of experiments and each lasts around 20 seconds to allow the vehicle to arrive at the traffic cone placement position and finish executing the driving decision.

Results and demo videos. The results show that our adversarial traffic cone can *always* fool the Apollo system in the entire trip across the 100 runs, leading to a *100%* vehicle collision rate. We inspect the experiment log and find that the adversarial cone evades both the camera and LiDAR perception pipelines at *every* frames before fusion, which thus fundamentally defeats the basic design assumption of using

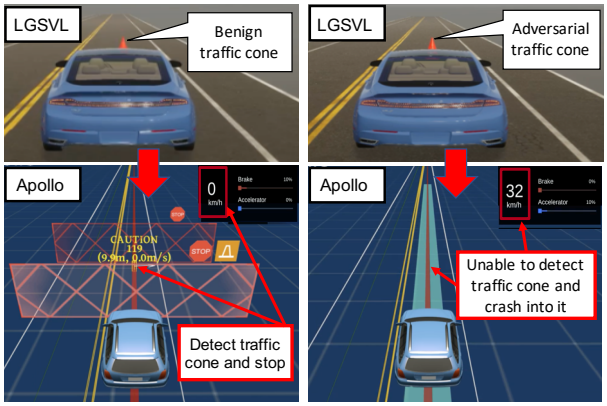


Figure 11: Screenshots of Apollo and LGSVL in the end-to-end attack evaluation with benign and adversarial traffic cones. Across 100 runs, the crash rate is 100% for adversarial case, and 0% for benign case.

MSF for defense. In contrast, in the benign case, Apollo is *always* able to correctly detect the benign cone and stop in front of it to avoid collision (i.e., 0% crash rate). Across different runs, the vehicle driving trajectories differ slightly due to the simulation randomness and sensor messaging delay/dropping, but our attack shows a high robustness against such trajectory variances when the victim is approaching.

Fig. 11 shows the key screenshots on both the LGSVL and Apollo sides during the simulation. As shown, in the benign case, the victim can detect the traffic cone and successfully make a stop decision to decrease its speed to 0 km/h. However, in the adversarial case, the victim cannot detect the traffic cone even when it is right in front of it. Thus, it maintains the original speed and directly crashes into it. We also record short demo videos from the simulation, available at [37].

VII. Limitations and Defense Discussion

A. Limitations of Our Study

End-to-end physical-world evaluation. In this work, our attack is designed with a practical attack model (§IV-A) and evaluated on real-world driving dataset and miniature-scale physical-world settings (§V-E). However, we did not perform an end-to-end attack evaluation on a real AV in the physical world due to the cost and safety considerations. As a best effort, we evaluate such end-to-end attack impacts using a production-grade AD simulator (§VI). Note that AD companies such as Waymo also heavily rely on simulation-based evaluations when developing and testing AD systems for safety and budget considerations [99].

Attack generality evaluation. In our evaluation, we target the MSF algorithms used in representative industry-grade AD systems such as Baidu Apollo [15], which generally adopt a rule-based fusion design. As introduced in §II-A, there also exists another type of fusion design: DNN-based fusion [18]–[24]. Thus, it is still unclear how effective MSF-ADV can be for DNN-based MSF algorithms. Note that this is not a limitation of our attack methodology: as described in §IV-E, our design is generally applicable to both fusion designs. Also, since rule-based fusion design is more preferable for the system development in the industry (§V-A), our current evaluation results can potentially lead to more impacts to AD systems in practice. Thus, we left the evaluation of MSF algorithms with DNN-based fusion as future work.

B. Defense Discussion

1) DNN-Level Defense: Our attack exploits vulnerability in DNNs used in MSF, and thus a direct defense direction is to secure these DNNs. In the recent arms race between adversarial attacks and defenses, various defense/mitigation techniques have been proposed, e.g., input transformation [100]–[102], adversarial training [65], and certified robustness [103], [104]. However, almost all of them focus on image classification models under digital-space attacks, instead of object detection models under physical-world attacks. To the best of our knowledge, no prior works has considered defending against adversarial 3D objects in MSF context.

Experiment methodology. In this case, as a best effort to understand the effectiveness of existing defenses in our attack setting, we perform experiments mainly on two easily-adaptable defense strategies: (1) camera/LiDAR input transformation without model re-training, for which we evaluate 4 popular methods: bit-depth reduction [100], median smoothing [100], JPEG compression [102], and autoencoder reformation [101]; and (2) augmenting training data, denoted as *AUG*, which re-trains the model with adversarial inputs mixed in training dataset [12], [50], [105]. AUG is only applied to YOLO v3 (Y3) since Apollo does not release training dataset for its models. Additionally, we also explored adversarial training [106] for Y3, but different from the standard adversarial training, we only applied *2 steps PGD attack* to approximate the solutions of inner maximal problem for efficiency due to the complexity of our attack pipeline (e.g., rendering, pre-processing, and attacking two models together) caused by our problem settings and evaluated system. Such a strategy has been adopted in some recent works to improve efficiency of adversarial training [106], [107]. More details are in Appendix E. Note that we do not evaluate certified robustness [103], [104] since its designs today focus on small 2D digital-space perturbations (e.g. ℓ_2=0.5 on ImageNet [108], [109]), and their extensions to either 3D space or physical-world attacks are still open research problems.

Results. Fig. 12 shows the results for the 4 input transformation based defenses on our attack on A5-L⊕A5-C for traffic cone. For each method, we explore different parameters to explore the trade-off between benign detection rate and attack success rate. As shown, with the decrease of the LiDAR/camera input quality (left to right for all the x-axes), the attack success rate will eventually increase for all 4 methods since the input quality becomes so low that both camera and LiDAR models cannot detect the object even in the benign case. For some methods, the attack success rate first decreases before such increase, which is likely because the input quality reduction disrupts our adversarial shape perturbations. Overall, median smoothing achieves the highest defense effectiveness

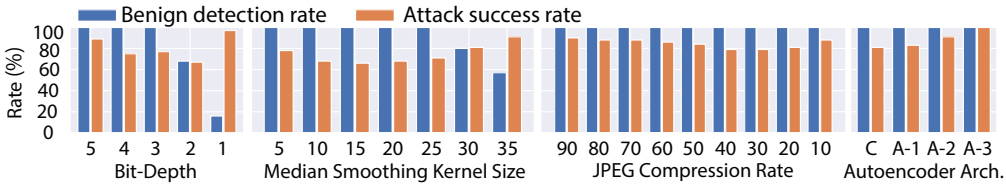


Figure 12. Evaluation results of 4 DNN input transformation based defense methods for our attack on A5-L⊕A5-C with traffic cone object. Benign detection rates mean detected by either LiDAR or camera. Attack success rate means that both LiDAR and camera fail to detect. For all x-axes, values from left to right mean higher to lower camera/LiDAR input quality (e.g., more smoothing or compression). Detailed setup in Appendix E.

	Y3 test set mAP	Benign det. rate	Attack succ. rate
Original	44%	100%	100%
AUG	32%	100%	69%

Table VII. Results of augmenting training data (AUG) for our attack on A5-L⊕Y3 compared to original model for bench object. Detailed setup in Appendix E.

by decreasing the attack success rate to 66% without affecting the benign detection rate. Note that it is known that all these methods can be bypassed by adaptive attacks [110]–[113]. Thus, an interesting future work is to explore the effectiveness of these methods under adaptive attack designs of MSF-ADV.

Table VII shows the results for AUG. For a fair comparison, the original model in the table is also newly-trained using the same setup. As shown, AUG is able to decrease the attack success rate to 69% with 100% benign detection rate. Our preliminary exploration of adversarial training with 2-step PGD does not show higher effectiveness: even with 900 epoch of training, the attack success rate is only reduced to 95% with 100% benign detection rate. The potential reason of the lower effectiveness is that 2 steps PGD is not enough to generate effective adversarial objects during training. Compared to some prior works [106], [107], this suggests that our attack poses more challenges in balancing the trade-off between efficiency and effectiveness in adversarial training. We plan to systematically investigate this in the future.

Overall, the most effective defense found in these experiments can only decrease the attack success rate to 66%, which is not quite enough to render this attack vector practically unexploitable. Leveraging the analysis insights, we plan to explore more effective defense designs by exploring (1) other input transformation considering the success of medium smoothing, and (2) more efficient and effective adversarial training designs for our attack. As certified robustness can provide strong theoretical guarantees, we also plan to explore the extensions of it to 3D space and physical-world attacks.

2) Fuse More Perception Sources: At MSF algorithm level, one defense direction is to fuse more perception sources, e.g., more cameras/LiDARs sharing an overlapped view but mounted at different positions, assuming that our attack may be more difficult to optimize if the fused camera/LiDAR perception results are from very different viewing angles and positions. Also, we may consider including RADAR into MSF, which is less preferred in state-of-the-art MSF designs (§II-A) but may help improve their security. Note that this cannot fundamentally defeat our attack since RADAR point clouds may also be affected by shape manipulations and their state-of-the-art object detection algorithms are still DNN-based [114]. Nevertheless, including RADAR may make it more difficult to attack if the RADAR perception model is more robust. We leave a systematic exploration of these to future work.

VIII. Related Work

Autonomous Driving (AD) system security. Since AD systems heavily rely on sensors, prior works have studied *sensor attacks* in AD context such as spoofing/jamming attacks on camera [40], [115], LiDAR [10], [29], RADAR [40], ultrasonic [40], and IMU [116]. In comparison, these works mainly focus on vulnerabilities at sensor level, while we focus on those at the higher *autonomy software* level, i.e., the "brain" of AD systems. At such level, prior works have studied the security of camera/LiDAR object detection [5], [6], [9], [10], [117] and tracking [118], localization [119], lane detection [120]–[122], traffic light detection [123], and end-to-end AD [12], [13]. However, so far *all* of them only consider attacks on camera or LiDAR perception *alone*, while we are the first to study the security of MSF-based AD perception and address the corresponding design challenges (§III-B).

Adversarial attacks. Various adversarial attacks have been proposed to generate adversarial attacks in the digital space [12], [50]–[56], [66], [85], [105], [124]–[127]. In comparison, we focus on physical-world attack vectors. Multiple prior works have designed and evaluated adversarial attacks in the physical world [5]–[9], [57]–[60]. However, none of them have considered MSF-based AD perception, and as described in §III-B, blindly combining their designs cannot directly achieve our goal due to various unique design challenges.

IX. Conclusion

This paper presents a first study on the security issues of MSF-based AD perception, that challenges the basic design assumption for MSF as a defense strategy in AD context. We design a novel attack method, MSF-ADV, with adversarial 3D object as the attack vector, and address design challenges in non-differentiable target camera and LiDAR sensing systems and non-differentiable computation of cell-level aggregated features for LiDAR. We perform evaluations on MSF algorithms included in industry-grade AD systems using real-world driving scenarios. Our results show that our attack achieves over 90% success rates across different object types and MSF algorithms, while being stealthy, robust, transferable and physical-world realizable. In simulation evaluation, our attack can cause 100% vehicle collision rate. We also evaluate and discuss defenses. Considering the critical role of perception for safe AV driving, we hope that our findings and insights can help the community develop effective defenses in practice.

ACKNOWLDGMENTS

We would like to thank Nicolas Papernot, Junjie Shen, Ziwen Wan, Takami Sato, Junze Liu, Xinyang Zhang, Xi Lu, Yu Stephanie Sun, Joshua Garcia, Pengchuan Zhang, Hongge Chen, Dan Luo, Benjamin Emerson Dolan, and the anonymous reviewers for valuable feedback on our work. This research was supported in part by the NSF under grants CNS-1850533, CNS-1929771, CNS-1932464, and CNS-2012001, USDOT under Grant 69A3552047138 for the CARMEN UTC, the ARO under contract W911NF1810208, and grant from Open Philanthropy and Good Ventures Foundation.

REFERENCES

[1] S. O.-R. A. V. S. Committee *et al.*, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," *SAE International: Warrendale, PA, USA*, 2018.

[2] "40+ Corporations Working On Autonomous Vehicles," https://www.cbinsights.com/research/autonomous-driverless-vehicl es-corporations-list.

[3] "Waymo has launched its commercial self-driving service in Phoenix - and it's called 'Waymo One'," https://www.businessinsider.com/waymo-one-driverless-car-service-launches-in-phoenix-arizona-2018-12.

[4] "UPS joins race for future of delivery services by investing in self-driving trucks," https://abcnews.go.com/Business/ups-joins-race-futur e-delivery-services-investing-driving/story?id=65014414.

[5] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, and D. Song, "Physical Adversarial Examples for Object Detectors," in *WOOT*, 2018.

[6] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen, "Seeing isn't Believing: Practical Adversarial Attack Against Object Detectors," *ACM CCS*, 2019.

[7] J. Lu, H. Sibai, and E. Fabry, "Adversarial Examples that Fool Detectors," *arXiv preprint arXiv:1712.02494*, 2017.

[8] Y. Zhang, P. D. Hassan Foroosh, and B. Gong, "CAMOU: Learning A Vehicle Camouflage For Physical Adversarial Attack On Object Detections In The Wild," in *ICLR*, 2019.

[9] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector," in *ECML PKDD*, 2018.

[10] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving," in *ACM CCS*, 2019.

[11] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards Robust LiDAR-based Perception in Autonomous Driving: General Black-box Adversarial Sensor Attack and Countermeasures," in *Usenix Security*, 2020.

[12] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated White-box Testing of Deep Learning Systems," in *SOSP*, 2017, pp. 1–18.

[13] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars," in *ICSE*, 2018.

[14] "Waymo Tech," https://waymo.com/tech/.

[15] "Baidu Apollo," http://apollo.auto.

[16] "Autoware.AI," https://www.autoware.ai//.

[17] "Pony.ai Tech," https://www.pony.ai/en/tech.html.

[18] D. Frossard and R. Urtasun, "End-to-end Learning of Multi-sensor 3D Tracking by Detection," in *ICRA 2018*. IEEE, 2018, pp. 635–642.

[19] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep Continuous Fusion for Multi-Sensor 3D Object Detection," in *ECCV*, 2018.

[20] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-View 3D Object Detection Network for Autonomous Driving," in *CVPR*, 2017.

[21] D. Xu, D. Anguelov, and A. Jain, "PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation," in *CVPR*, 2018, pp. 244–253.

[22] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-Task Multi-Sensor Fusion for 3D Object Detection," in *CVPR*, 2019.

[23] X. Du, M. H. Ang, and D. Rus, "Car Detection for Autonomous Vehicle: LIDAR and Vision Fusion Approach Through Deep Learning Framework," in *IROS*, 2017.

[24] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," in *IROS*, 2018, pp. 1–8.

[25] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, "Accurate Monocular 3D Object Detection via Color-Embedded 3D Reconstruction for Autonomous Driving," in *CVPR*, 2019, pp. 6851–6860.

[26] X. Du, M. H. Ang, S. Karaman, and D. Rus, "A General Pipeline for 3D Detection of Vehicles," in *ICRA 2018*. IEEE, 2018, pp. 3194–3200.

[27] R. Quinonez, J. Giraldo, L. Salazar, and E. Bauman, "SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants," in *USENIX Security*, 2020.

[28] P. Guo, H. Kim, N. Virani, J. Xu, M. Zhu, and P. Liu, "RoboADS: Anomaly Detection against Sensor and Actuator Misbehaviors in Mobile Robots," in *DSN*. IEEE, 2018, pp. 574–585.

[29] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and Dazzle: Adversarial Optical Channel Exploits against LiDARs for Automotive Applications," in *CHES*. Springer, 2017, pp. 445–467.

[30] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast Object Detection in 3D Point Clouds using Efficient Convolutional Neural Networks," in *ICRA*, 2017.

[31] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *CVPR*, 2018, pp. 4490–4499.

[32] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast Encoders for Object Detection from Point Clouds," in *CVPR*, 2019, pp. 12 697–12 705.

[33] J. Beltran, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera, "Birdnet: a 3D Object Detection Framework from Lidar Information," in *ITSC*. IEEE, 2018, pp. 3517–3523.

[34] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *CVPR 2018*, 2018, pp. 7652–7660.

[35] "Navigant Research Names Waymo, Ford Autonomous Vehicles, Cruise, and Baidu the Leading Developers of Automated Driving Systems," https://www.businesswire.com/news/home/20200407005119/en /Navigant-Research-Names-Waymo-Ford-Autonomous-Vehicles.

[36] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KiTTi Dataset," *IJRR*, 2013.

[37] "Our Project Website," https://sites.google.com/view/cav-sec/msf-adv.

[38] R. Ivanov, M. Pajic, and I. Lee, "Attack-resilient Sensor Fusion," in *DATE*. IEEE, 2014, pp. 1–6.

[39] W. Xu, C. Yan, W. Jia, X. Ji, and J. Liu, "Analyzing and Enhancing the Security of Ultrasonic Sensors for Autonomous Vehicles," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5015–5029, 2018.

[40] C. Yan, W. Xu, and J. Liu, "Can You Trust Autonomous Vehicles: Contactless Attacks against Sensors of Self-driving Vehicle," *DEF CON*, vol. 24, no. 8, p. 109, 2016.

[41] J. Petit and S. E. Shladover, "Potential Cyberattacks on Automated Vehicles," *IEEE ITS*, vol. 16, no. 2, pp. 546–556, 2014.

[42] Y. Man, M. Li, and R. Gerdes, "GhostImage: Perception Domain Attacks against Vision-based Object Classification Systems," *arXiv preprint arXiv:2001.07792*, 2020.

[43] V. Chandrasekaran, B. Tang, N. Papernot, K. Fawaz, S. Jha, and X. Wu, "Rearchitecting Classification Frameworks For Increased Robustness," in *arXiv:1905.10900*, 2019.

[44] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," *IEEE NNLS*, pp. 3212–3232, 2019.

[45] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[46] L. Chi and Y. Mu, "Deep Steering: Learning End-to-End Driving Model from Spatial and Temporal Visual Cues," *arXiv*, 2017.

[47] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep Learning for 3D Point Clouds: A Survey," 2019.

[48] "Velodyne Alpha Prime," https://autonomoustuff.com/product/velodyn e-vls-128/.

[49] D. Z. Wang and I. Posner, "Voting for Voting in Online Point Cloud Object Detection," in *Robotics: Science and Systems*, 2015.

[50] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *ICLR*, 2015.

[51] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," in *Euro S&P*, 2016.

[52] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating Adversarial Examples with Adversarial Networks," *ArXiv*, 2018.

[53] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially Transformed Adversarial Examples," *ICLR*, 2018.

[54] C. Xiao, R. Deng, B. Li, F. Yu, M. Liu, and D. Song, "Characterizing Adversarial Examples based on Spatial Consistency Information for Semantic Segmentation," in *ECCV*, 2018, pp. 217–234.

[55] C. Xiao, X. Pan, W. He, J. Peng, M. Sun, J. Yi, M. Liu, B. Li, and D. Song, "Characterizing Attacks on Deep Reinforcement Learning," *arXiv*, 2019.

[56] H. Qiu, C. Xiao, L. Yang, X. Yan, H. Lee, and B. Li, "SemanticAdv: Generating Adversarial Examples via Attribute-conditioned Image Editing," in *ECCV*. Springer, 2020, pp. 19–37.

[57] J. Li, F. Schmidt, and Z. Kolter, "Adversarial Camera Stickers: A Physical Camera-based Attack on Deep Learning Systems," in *ICML*, 2019, pp. 3896–3904.

[58] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial Patch," in *arXiv:1712.09665*, 2017.

[59] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling Automated Surveillance Cameras: Adversarial Patches to Attack Person Detection," in *CVPR Workshops*, 2019, pp. 0–0.

[60] A. Athalye and I. Sutskever, "Synthesizing Robust Adversarial Examples," in *International Conference on Machine Learning (ICML)*, 2018.

[61] "Does your car have automated emergency braking? It's a big fail for pedestrians," https://www.zdnet.com/article/does-your-car-have-automated-emergency-braking-its-a-big-fail-for-pedestrians/, 2019.

[62] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust Physical-World Attacks on Deep Learning Visual Classification," in *CVPR*, 2018.

[63] "Avis will Service Waymo's Self-driving Minivans," https://www.theverge.com/2017/6/26/15873236/avis-waymo-google-self-driving-cars-vans.

[64] "Experimental Security Research of Tesla Autopilot," https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf.

[65] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," in *ICLR*, 2018.

[66] N. Carlini and D. A. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *IEEE S&P*, 2017.

[67] "3D Printing Online," https://formlabs.com/software/.

[68] K. Eykholt, "Designing and Evaluating Physical Adversarial Attacks and Defenses for Machine Learning Algorithms," Ph.D. dissertation, 2019.

[69] B. Huang and H. Ling, "SPAA: Stealthy Projector-based Adversarial Attacks on Deep Image Classifiers," *ArXiv*, 2020.

[70] "Traffic Cone," https://en.wikipedia.org/wiki/Traffic_cone.

[71] "Intro to Rendering, Ray Casting," https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/MIT6_837F12_Lec11.pdf.

[72] H. Kato, Y. Ushiku, and T. Harada, "Neural 3D Mesh Renderer," in *CVPR*, June 2018.

[73] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *IROS*, 2015, pp. 922–928.

[74] "Tri. Interpolation," en.wikipedia.org/wiki/Trilinear_interpolation.

[75] S. Li, X. Xu, L. Nie, and T.-S. Chua, "Laplacian-Steered Neural Style Transfer," in *ICMR*, 2017, pp. 1716–1724.

[76] "Brake Distance," http://www.csgnetwork.com/stopdistcalc.html.

[77] "Carma Platform," https://github.com/usdot-fhwa-stol/carma-platform.

[78] "Autoware Self-driving Vehicle on a Highway," https://www.youtube.com/watch?v=npQMzH3j_d8.

[79] "Baidu launches their open platform for autonomous cars–and we got to test it," https://technode.com/2017/07/05/baidu-apollo-1-0-autonomous-cars-we-test-it/.

[80] "Baidu Launches Public Robotaxi Trial Operation," https://www.globenewswire.com/news-release/2019/09/26/1921380/0/en/Baidu-Launches-Public-Robotaxi-Trial-Operation.html.

[81] "YOLOv3 Darknet," https://pjreddie.com/darknet/yolo/.

[82] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *CVPR*, 2018.

[83] Y. Jo, S. Yang, and S. Joo Kim, "Investigating Loss Functions for Extreme Super-Resolution," in *CVPR Workshops*, 2020, pp. 424–425.

[84] "Amazon Mechanical Turk," https://www.mturk.com.

[85] T. Tsai, K. Yang, T.-Y. Ho, and Y. Jin, "Robust Adversarial Objects against Deep Learning Models," in *AAAI*, 2020.

[86] "FormLabs," https://formlabs.com/software/.

[87] "Curvature," https://en.wikipedia.org/wiki/Gaussian_curvature.

[88] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[89] "Adding Gaussian Noise," https://pytorch.org/docs/stable/tensors.html.

[90] "Genetic Algorithm," https://pypi.org/project/geneticalgorithm/.

[91] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, "Genattack: Practical Black-box Attacks with Gradient-free Optimization," in *GECCO*, 2019.

[92] Y. Feng, B. Wu, Y. Fan, L. Liu, Z. Li, and S. Xia, "CG-ATTACK: Modeling the Conditional Distribution of Adversarial Perturbations to Boost Black-Box Attack," 2020.

[93] R. Alika, E. M. Mellouli, and E. H. Tissir, "Optimization of Higher-Order Sliding Mode Control Parameter using Particle Swarm Optimization for Lateral Dynamics of Autonomous Vehicles," in *IRASET*. IEEE, 2020, pp. 1–6.

[94] S. Dominguez, A. Ali, G. Garcia, and P. Martinet, "Comparison of Lateral Controllers for Autonomous Vehicle: Experimental Results," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1418–1423.

[95] "LARGE-FORMAT 3D PRINTER FOR INDUSTRIAL APPLICATIONS," https://bigrep.com/bigrep-one/.

[96] AASHTO, *Policy on Geometric Design of Highways and Streets (7th Edition)*, 2018. [Online]. Available: https://app.knovel.com/hotlink/toc/id:kpPGDHSE12/policy-geometric-design/policy-geometric-design

[97] "Apollo Models," https://github.com/ApolloAuto/apollo/tree/r5.5.0/modules/perception/production/data/perception/lidar/models/cnnseg.

[98] "LGSVL Simulator," https://www.lgsvlsimulator.com/.

[99] "Inside Waymo's Secret World for Training Self-Driving Cars," https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/.

[100] W. Xu, D. Evans, and Y. Qi, "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks," in *NDSS*, 2018.

[101] D. Meng and H. Chen, "MagNet: a Two-Pronged Defense against Adversarial Examples," in *ACM CCS*, 2017, pp. 135–147.

[102] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A Study of the Effect of JPG Compression on Adversarial Images," *arXiv*, 2016.

[103] L. Li, X. Qi, T. Xie, and B. Li, "SoK: Certified Robustness for Deep Neural Networks," *arXiv preprint arXiv:2009.04131*, 2020.

[104] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified Robustness to Adversarial Examples with Differential Privacy," in *IEEE S&P*.

[105] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing Properties of Neural Networks," in *ICLR*, 2014.

[106] H. Zhang and J. Wang, "Towards Adversarially Robust Object Detection," in *ICCV*, 2019, pp. 421–430.

[107] E. Wong, L. Rice, and J. Z. Kolter, "Fast is Better than Free: Revisiting Adversarial Training," *ICLR*, 2020.

[108] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified Adversarial Robustness via Randomized Smoothing," *ICML*, 2019.

[109] G. Yang, T. Duan, E. Hu, H. Salman, I. Razenshteyn, and J. Li, "Randomized Smoothing of All Shapes and Sizes," *ICML*, 2020.

[110] N. Carlini and D. Wagner, "MagNet and "Efficient Defenses against Adversarial Attacks" are not Robust to Adversarial Examples," *arXiv*, 2017.

[111] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang, "Interpretable Deep Learning under Fire," in *USENIX Security*, 2020.

[112] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial Example Defense: Ensembles of Weak Defenses are not Strong," in *USENIX WOOT*, 2017.

[113] Y. Sharma and P.-Y. Chen, "Bypassing Feature Squeezing by Increasing Adversary Strength," *ICLR Workshop*, 2018.

[114] L. Wang, J. Tang, and Q. Liao, "A Study on Radar Target Detection based on Deep Neural Networks," *IEEE Sensors Letters*, pp. 1–4, 2019.

[115] B. Nassi, D. Nassi, R. Ben-Netanel, Y. Mirsky, O. Drokin, and Y. Elovici, "Phantom of the ADAS: Phantom Attacks on Driver-Assistance Systems," in *IACR*, 2020.

[116] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors," in *USENIX Security*, 2018, pp. 1545–1562.

[117] "Model Hacking ADAS to Pave Safer Roads for Autonomous Vehicles," https://www.mcafee.com/blogs/other-blogs/mcafee-labs/model-hacking-adas-to-pave-safer-roads-for-autonomous-vehicles/, 2020.

[118] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. Wei, "Fooling Detection Alone is Not Enough: Adversarial Attack Against Multiple Object Tracking," in *ICLR*, 2019.

[119] J. Shen, J. Y. Won, and Q. A. Chen, "Drift with Devil: Security of Multi-Sensor Fusion based Localization in High-Level Autonomous Driving under GPS Spoofing," in *Usenix Security*, 2020.
[120] T. Sato, J. Shen, N. Wang, Y. J. Jia, X. Lin, and Q. A. Chen, "Hold Tight and Never Let Go: Security of Deep Learning based Automated Lane Centering under Physical-World Attack," *ArXiv*, 2020.
[121] T. Sato, J. Shen, N. Wang, Y. J. Jia, X. Lin, and Q. A. Chen, "Deployability Improvement, Stealthiness User Study, and Safety Impact Assessment on Real Vehicle for Dirty Road Patch Attack," in *AutoSec Workshop at NDSS*, 2021.
[122] H. Liang, R. Jiao, T. Sato, J. Shen, Q. A. Chen, and Q. Zhu, "End-to-End Analysis of Adversarial Attacks to Automated Lane Centering Systems," in *AutoSec Workshop at NDSS*, 2021.
[123] K. Tang, J. Shen, and Q. A. Chen, "Fooling Perception via Location: A Case of Region-of-Interest Attacks on Traffic Light Detection in Autonomous Driving," in *AutoSec Workshop at NDSS*, 2021.
[124] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a Simple and Accurate Method to Fool Deep Neural Networks," in *CVPR*, 2016, pp. 2574–2582.
[125] C. Xiao, D. Yang, B. Li, J. Deng, and M. Liu, "MeshAdv: Adversarial Meshes for Visual Recognition," in *IEEE CVPR*, 2019.
[126] C. Xiang, C. R. Qi, and B. Li, "Generating 3D Adversarial Point Clouds," in *CVPR*, 2019, pp. 9136–9144.
[127] K. Lee, Z. Chen, X. Yan, R. Urtasun, and E. Yumer, "ShapeAdv: Generating Shape-Aware Adversarial 3D Point Clouds," *ArXiv*, 2020.
[128] "Understanding Accuracy, Precision, and Tolerance in 3D Printing," https://formlabs.com/blog/understanding-accuracy-precision-tolerance-in-3d-printing/.
[129] "User Study: Anomalous Traffic Cone Survey," https://drive.google.com/file/d/1EqtQL6m1ZPNOQGs6pbAM25WFT8D58EC2/view.
[130] "Mesh Simplification," http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/08_Simplification.pdf.
[131] "Pillow (PIL Fork)," https://pillow.readthedocs.io/en/stable/.
[132] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial Training for Free!" in *NIPS*, 2019, pp. 3358–3369.
[133] D. Hendrycks, K. Lee, and M. Mazeika, "Using Pre-Training can Improve Model Robustness and Uncertainty," *ICML*, 2019.
[134] T. Chen, S. Liu, S. Chang, Y. Cheng, L. Amini, and Z. Wang, "Adversarial Robustness: From Self-Supervised Pre-Training to Fine-Tuning," in *CVPR*, 2020, pp. 699–708.
[135] "COCO Dataset," http://cocodataset.org/.

APPENDIX

Parameter	Value
PGD initial point (§IV-E)	0.01
PGD constraint (§IV-E)	0.02
Tanh approximation parameter μ (§IV-D)	100
Preventing division by zero ε (§IV-D)	10^{-7}
X sample range (§IV-E)	$(5, 35)$
Y sample range (§IV-E)	$(-0.3, 0.3)$
yaw sample angles (§IV-E)	$(-5°, 5°)$
Learning rate (§IV-E)	0.001
$\mathcal{L}_r(\cdot)$ coefficient λ (§IV-E)	20
Height loss coefficient β_1 (Appendix A)	0.001
Precision of 3D printer used in §V-E	0.38mm

Table VIII. Detailed settings for attack parameters in §V.

A. Realizability loss $\mathcal{L}_r(\cdot)$ in §IV-E

To realize our attack goal in §III-A, S^a needs to be 3D-printed and placed on top of the road surface in the physical world. To facilitate this, we design the realizability loss $\mathcal{L}_r(\cdot)$ in our objective function to (1) improve the printability of S^a by 3D printers, and (2) prevent the generation of S^a that is underneath the road surface. Our formulation of $\mathcal{L}_r(\cdot)$ is in Eq. (9), where the first and second parts are for achieving (1) and (2) respectively. The first part is a Laplacian loss [75], where V^a is the vertex set of S^a, and for $\boldsymbol{v}_i^a \in V^a$, $\Gamma(\boldsymbol{v}_i^a)$ denotes the set of connected neighboring vertices of $\boldsymbol{v}_i^a$. Since our attack generation is performed by only moving the vertex positions in the benign object S (§IV-A), there is always a corresponding vertex $\boldsymbol{v}_i$ in the vertex set V of S that $\boldsymbol{v}_i^a$ is moved from. The distance between $\boldsymbol{v}_i^a$ and $\boldsymbol{v}_i$ is denoted as $\Delta\boldsymbol{v} = \boldsymbol{v}_i^a - \boldsymbol{v}_i$. Thus, the first part in Eq. (9) penalties the differences between the position change of each vertex in S^a and those of its neighboring vertices. This can thus improve the smoothness of the surface of S^a, which can lower the precision requirements of the 3D printer [128] and thus improve the printability of S^a. We also use a popular mesh simplification method, Quadric Edge Collapse Decimation (QECD), as an optional post-processing step to further improve printability.

In the second part, z_i^a and z_i denotes the height values of $\boldsymbol{v}_i^a$ and $\boldsymbol{v}_i$. This part minimizes the distance between the lowest height among all vertices in S^a and that in S, which thus penalties the moving of the vertices in S^a towards under the road surface. β_1 is a hyper-parameter for this part in Eq. (9).

$$\mathcal{L}_r(S^a, S) = \sum_{\boldsymbol{v}_i^a \in V^a} \sum_{\boldsymbol{v}_q^a \in \Gamma(\boldsymbol{v}_i^a)} \left\| \Delta\boldsymbol{v}_i^a - \Delta\boldsymbol{v}_q^a \right\|_2^2 + \beta_1 \cdot \| \min_{\boldsymbol{v}_i^a \in V^a} z_i^a - \min_{\boldsymbol{v}_i \in V} z_i \|_2^2 \tag{9}$$

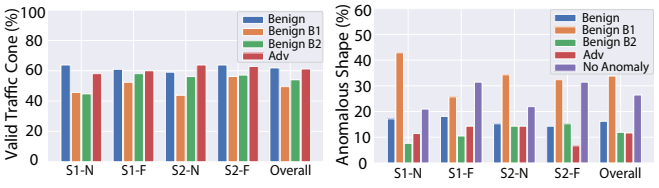
B. Attack Stealthiness User Study

In this section, we conduct a user study to evaluate the stealthiness of the adversarial 3D objects. We go through the IRB process and our study is determined as the IRB Exempt, due to not involving collection of any Personally Identifiable Information (PII) or target any sensitive population.

Evaluation methodology. In this study, we select traffic cone as the evaluation target due to its high attractiveness for the attacker (§V-A). We evaluate 4 red traffic cone with different shapes: the benign shape (*Benign*) the adversarial shape generated by MSF-ADV (*Adv*), and two benign but broken shapes similar to Fig. 2 (*Benign B1* and *B2*). We consider *Benign B1* and *B2* since our attack is designed to mimic benign traffic objects with a broken look (§IV-A). We randomly select two images (*S1*, *S2*) from KITTI and render these shapes into these two images at two different positions (near or far way from the victim AV, denoted as *N* or *F*) to generate four realistic driver's scenario (*S1-N*, *S1-F*, *S2-N*, *S2-F*) on the roadway for each shape.

For each of the 4 rendered images above, we ask whether the red traffic object in the image is a valid traffic cone. Note that the driving images are selected by ensuring no extra red object. Images of them are on our website [37]. Among the 4 rendered images, we also ask in which one the red traffic object has the most anomalous shape compared to a normal traffic cone. "No Anomaly" option is included to avoid randomly picking from the participants. To understand the distribution of the participant's background, we also ask for demographic information and background information related to driving.

Evaluation setup. We use Amazon Mechanical Turk [84] to perform the user study. In total, we collected results from



(a) Validity of the traffic cone (b) Most anomalous shape

Figure 13: User study results of the attack stealthiness. (a) shows the ratio of users thinking a given object is a valid traffic cone; (b) shows the selection ratios for traffic cones with the most anomalous shape. *S1* and *S2*: 2 different real-world driving images; *N* and *F*: near and far rendering positions.

105 participants (55.24% male and 44.76% female) with 35.3 average age. We confirmed that all of them have driving experience by asking them the age when first licensed and the weekly driving mileage. All the benign objects, including *Benign B1* and *B2* can be correctly detected by the latest Apollo MSF combination (A5-L⊕A5-C) while *Adv* cannot. The full survey is available at [129].

Results. Fig. 13 (a) shows the ratio of users thinking that the given traffic cone object is a valid traffic cone. As shown, *Benign* and *Adv* have similar ratios (around 60%) and are higher than *Benign B1* and *B2* since the broken shapes may be more obvious than that of *Adv* after our surface smoothing and PGD-based perturbation bounding (§IV-E). Note that even for *Benign* there are around 40% users thinking that it is invalid. This might be because the rendered color and shading inevitably have infidelity compared to the real-world background images. Fig. 13 (b) shows the selection ratios for the cone object with the most anomalous shape. As shown, *Benign B1* and "No Anomaly" are the most popular choices across and *Adv* is always the lowest. The results show that our generated adversarial traffic cone is generally viewed at least as innocent as the original benign cone, and also less suspicious than certain benign ones with broken shapes.

C. Attack Effectiveness under Different Attack Settings

In this section, we perform experiments to understand how sensitive our attack is to different attack parameter settings.

Experimental setup. We target 5 key attack parameters in Table VIII to perform experiments: μ, λ, β_1, Learning rate, and PGD initial. For each parameter, we experiment with values that are one magnitude higher or lower than the default value. The experiments are performed on A5-L⊕A5-C with traffic cone. The results are averaged over 20 attack scenarios randomly selected from the 100 scenarios in §V-A.

Results. Table IX shows the results. As shown, our attack is most sensitive to μ. Considering that μ is used in our differentiable approximation of the point-inclusion calculation (Eq. (8) in §IV-D), these results show that our differentiable approximation design is critical to the attack success. Different learning rates, λ, and PGD initial are also shown to impact the results, but such impacts are limited to when the values are above or below a certain magnitude.

D. Printability Evaluation

To perform our attack, the adversarial objects generated digitally need to be (1) printable by today's 3D printers, and (2) the easier to be printed the better, e.g., requiring less printing precision and thus printable by cheaper 3D printers. In this section, we evaluate such printability of our attack.

Evaluation metrics. To evaluate whether it is *printable or not*, we first use *PreForm*, a commercial printability checking tool [86] that can determine whether their 3D-printing service can print a given 3D mesh. We also leverage the object *watertightness* [85] as another metric, which measures whether the object mesh could hold water if filled. Thus, any 3D object needs to be watertight to have a volume and thus can validly exist (and thus 3D-printed) in physical world. This is the most basic metric for any object meshes to be printable.

For whether the object is *easy to print*, we use the *self-intersection ratio* and *curvature*. *Self-intersection ratio* measures the percentage of the object mesh's 2D faces that have intersections with its other faces. High self-intersection ratio means the mesh need to be printed by a higher precision printer with higher cost. The second metric we use is the *curvature* of the object, which measures how smooth the object surface is. The more smooth the surface is, the less printing precision is required and thus the easier to print. We calculate this metric using the average per-vertex Gaussian curvature value.

Experimental setup. As described in §IV-E and Appendix A, our design includes two methods to improve the printability: the Laplacian loss (LP) in $\mathcal{L}_r(\cdot)$ in Eq. (9), and QECD [130] as an optional post-processing step. Thus, in our experiment we evaluate the printability of our adversarial objects with and without these two methods. We use the same scenario and parameter settings as §V-B.

Results. Table X shows the evaluation results for A5-L⊕A5-C using traffic cones. As shown, with or without using any printability improvement methods, the objects generated by our method are all watertight and determined as printable by the PreForm software since our attack method only manipulates the vertex positions of the benign object without changing the original vertex connection relationships.

For the two metrics on whether the object is easy to print, both the self-intersection ratio and the average curvature value are greatly reduced by applying either LP or QECD. LP alone is particularly cost-effective: it is able to substantially reduce the self-intersection ratio by 74.3% and the curvature value by 58.9% without hurting the attack success rate. However, QECD alone hurts self-intersect ratio, curvature value and attack success rate. The decrease of the attack success rate is because QECD is a mesh simplification method that may slightly change the object shape, which thus may interfere with the originally well-optimized shape of the adversarial object. Combining QECD and LP together achieves the highest reduction in both metrics, with only 0.46% self-intersection ratio and 0.67 curvature value. Note that, the average curvature for the benign traffic cone object is 0.72. Thus, both LP alone and the combination achieve a similar level of surface

	μ			λ			β_1			Learning rate			Initialization		
Key attack parameters	10	100	1000	2.0	20.0	200.0	0.01	0.001	0.0001	0.01	0.001	0.0001	0.1	0.01	0.001
Attack success rate	75%	100%	60%	100%	100%	65%	100%	100%	100%	75%	100.0%	100%	45%	100.0%	100%

Table IX. Attack success rate for A5-L⊕A5-C with traffic cone under different attack parameter settings. Descriptions of these attacker parameters are in Table VIII. Gray cells are default settings.

Technique used	Printable?		Easy to Print?		Success Rate
	PreForm	Watertight	Self-intersect	Curvature	
None	100%	100%	88.73%	1.68 ± 1.56	100%
LP	100%	100%	14.43%	0.69 ± 0.65	100%
QECD	100%	100%	38.96%	1.42 ± 1.30	90%
LP + QECD	100%	100%	0.46%	0.67 ± 0.50	92%

Table X. Printability evaluation results of MSF-ADV on A5-L⊕A5-C with traffic cone. LP: Laplacian loss in Eq. (9).

smoothness comparable to a normal real-world object, which thus are printable enough in practice. While the combination reduces the self-intersection ratio compared to LP alone, it incurs 8% success rate decrease due to the use of QECD. Thus, there exists a trade-off. If the attacker does not care about the printing cost, they can choose to use LP alone to better ensure the attack success; otherwise, they can combine it with QECD to reduce the printing costs.

E. Details of the DNN-Level Defenses Evaluated in §VII-B1

We describe the details of the defense methods.

Bit-Depth reduction [100]. We follow the setting in prior work [100]. We reduce the bit depth for the image input and the LiDAR point cloud. For a camera image, it consists of RGB channel with 8-bit depth (0-255) for each of them. For a LiDAR point cloud, each point has 4 fields: x, y, z, i, where x, y, and z represents the 3D position, and i is intensity. Each field is a floating point with 32-bit. We use the formulation: $\frac{\text{round}(x*(2^{\text{bit}}-1))}{(2^{\text{bit}}-1)}$ to reduce the bit-depth. In our experiments, we evaluate 5 different bit-depths ranging from 5-bits to 1-bit for both camera and LiDAR inputs. Higher bit-depth number means higher input quality after the bit-depth reduction.

Median smoothing [100]. We follow the setting in prior work [100] and apply the median smoothing to both LiDAR and camera inputs by taking a median around each LiDAR point or camera pixel with a different kernel size. We evaluate 7 different kernel sizes ranging from 5 to 35. Larger the kernel size means higher smoothness and lower input quality.

JPEG compression [102]. We follow the setting in [102] and apply the JPEG only to images since JPEG compression is specific to images. Our attack is successful only when it succeeds for both camera and LiDAR models, therefore, securing the camera model alone is still an effective defense strategy. We use Python Image Library (PIL) [131] to control the compression quality with argument "quality". We use 9 values from 10 to 90 with step 10 to explore the defense effectiveness at different compression rates. Lower values means higher compression rates and thus lower image quality.

Autoencoder reformation [101]. Autoencoder reformation is a part of MagNet defense [101]. We apply it only on image since it is designed for camera-based adversarial examples.

We evaluate 4 different autoencoder architectures, denoted as *C*, *A-1*, *A-2*, and *A-3*. C is the same architecture in the MagNet paper [101] for the CIFAR-10 dataset. Since the input size in our setting is much larger than that in CIFAR-10, we also evaluate 3 other architectures, A-1, A-2, and A-3, by adding 1, 2, 3 average pooling layers to C. From C, to A-3, the latent space dimension size decreases, which thus means more compression and lower input quality. All the autoencoder are trained with real-world images in KITTI dataset [36](§V).

Adversarial training (AT) [106]. Since Apollo does not release the training dataset for its models, we can only evaluate this method on Y3 (YOLO v3). Since our attack needs to succeed for both camera and LiDAR, a secure camera model is still an effective defense strategy. We adapt our method to the state-of-the-art adversarial training-based method for camera-based object detection [106]. We follow their algorithm but change the attack in the training loop to ours. Since our attack is performed by adding an object instead of perturbing an existing one, an additional challenge is how to assign ground-truth bounding boxes and labels to our adversarial objects. To address this, we render benign object to the same position and use its detection results as ground-truth results for adversarial one. In AT, we only use bench object to perform experiment.

While adversarial training can be highly robust, it is known to be expensive and nearly intractable for large-scale problems [107], [132]. In our case, this problem further exacerbates as the cost of our attack is higher than 2D digital-space attacks. Thus, we employ an acceleration method found in a recent work [107] that allows a much smaller number of PGD steps (instead of a full optimization cycle) in each training iteration by randomly initializing the adversarial inputs. Specifically, we use a PGD with 2 step and randomly initialize the adversarial mesh during each training iteration. Besides, we train our model from a pre-trained Y3 model, which can converge much faster and also improve robustness [133], [134]. We use the original Y3 training set COCO [135]. We train the model for over 900 epoch, and the model converges after ∼83 epoch.

Augmenting training data (AUG) [12], [50], [105]. Prior works show that re-training the model with adversarial inputs mixed in the original training data can improve the model robustness [12], [50], [105]. Same as for adversarial training, this method is only applied to Y3, and we use the same method as in adversarial training to generate the adversarial inputs and their ground-truth bounding boxes and labels. We use same COCO training dataset and the number of training epoch. In this case, the model converges at ∼48 epoch.